6

Managing Active Directory

In this chapter, we cover the following recipes:

- Installing an AD forest root domain
- Testing an AD installation
- Installing a replica domain controller
- Installing a child domain
- Creating and managing AD users and groups
- Managing AD computers
- Adding/removing users using CSV files
- Creating group policy objects
- Reporting on AD replication
- Reporting on AD computers
- Reporting on AD users

Introduction

A core component of almost all organizations' IT infrastructure is **Active Directory** (**AD**). AD provides access control, user and system customization, and a wealth of directory and other services. Microsoft first introduced AD with Windows 2000 and has improved and expanded the product with each successive release of Windows Server.

Over the years, Microsoft has made AD more of a brand than a single feature. At the core is **Active Directory Domain Services** (**AD DS**). There are four additional Windows Server features under the AD brand:

- **AD Certificate Services** (**AD-CS**): This allows you to issue X.509 certificates for your organization. For an overview of AD-CS, see `https://docs.microsoft.com/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831740(v=ws.11)`.
- **AD Federation Services** (**AD-FS**): This feature enables you to federate identity with other organizations to facilitate interworking. You can find an overview of AD-FS at `https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831502(v=ws.11)`.
- **AD Lightweight Directory Services** (**AD-LDS**): This provides rich directory services for use by applications. You can find an overview of AD-LDS at `https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831593(v=ws.11)`.
- **AD Rights Management Services** (**AD-RMS**): RMS enables you to control the rights to document access to limit information leakage. For an overview of RMS, see `https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh831364(v=ws.11)`.

Note that the overview documents referred to above are older documents based on Windows Server 2012. At the time of writing, the documentation teams have not updated them fully to reflect the latest Windows Server version. The overview and the essential operation of these features remain mostly unchanged.

Active Directory's domain service is complex, and there are a lot of moving parts. With AD, you have a logical structure consisting of forests, domains, domain trees, and **organizational units** (**OUs**). You also have the physical structure, including **domain controllers** (**DCs**) and **global catalogs** (**GCs**). There is also a replication mechanism to replicate objects across your domain. For a deeper look at the architecture of AD, see `https://activereach.net/support/knowledge-base/connectivity-networking/understanding-active-directory-its-architecture/`.

A **forest** is a top-level container that houses domains. A forest is a security boundary, although you can set up cross-forest trusts to enable interworking between multiple forests. AD bases forest (and domain) names on DNS. AD DCs and AD clients use DNS to find the IP address of DCs, which makes DNS a critical part of your AD infrastructure.

A **domain** is a collection of objects, including users, computers, policies, and much more. You create a forest by installing the forest's first domain controller. In AD domains, trees are collections of domains that you group in a hierarchical structure. Most organizations use a single domain (and domain tree) within a single forest. Multiple domains in one or more domain trees are also supported, but the best practice is to avoid them.

A **domain controller** is a Windows Server running AD and holding the objects for a given domain. ALL domains must have at least one DC, although best practice is always to have at least two. You install the AD DS service onto your server, then promote the server to be a DC.

The **global catalog** is a partial replica of objects from every domain in an object to enable searching. Exchange, for example, uses the GC heavily. You can have the GC service on some or all DCs in your forest. Generally, you install the GC facility while you promote a Windows Server to be a DC.

Using AD DS (or AD) and PowerShell, you can deploy your domain controllers throughout your organization. Use the *Installing an AD forest root domain* recipe to install a forest root domain controller and establish an AD forest.

Installing features and services using PowerShell in a domain environment often uses remoting, which in turn requires authentication. From one machine, you use PowerShell remoting to perform operations on other systems. You need the correct credentials for those operations.

In this book, you use a domain, `Reskit.Org`, for most of the recipes. In this chapter, you establish the forest and domain (and a child domain) and create users, groups, and organizational units which you rely on in later chapters.

Once you have a first DC in your Reskit.Org forest, you should add a replica DC to ensure reliable domain operations. In *Installing a replica domain controller,* you add a second DC to your domain. In *Installing a child domain*, you extend the forest and add a child domain to your forest.

Active Directory uses a database of objects that include users, computers, and groups. In the *Creating and managing AD users and groups* recipe, you create, move, and remove user and group objects and create and use organizational units. In *Managing AD computers*, you manage the computers in your AD, including joining workgroup systems to the domain. In the *Adding/removing users using CSV files* recipe, you add users to your AD using a **comma-separated values** (**CSV**) file containing users' details.

Group Policy is another important feature of AD. With group policy, you can define policies for users and computers that Windows applies automatically to the user and/or computer. In the *Creating group policy objects* recipe, you create a simple **group policy object** (**GPO**) and observe applying that policy.

Active Directory can make use of multiple DCs for both load balancing and fault tolerance. Such DCs must be synchronized whenever you make changes to your AD, and AD replication performs that function. In *Reporting on AD replication*, you examine tools to help you manage and troubleshoot replication.

In the recipes *Reporting on AD computers* and *Reporting on AD users*, you examine the AD to find details on computers that have not started up or logged onto the domain. You also look at user accounts for users who are members of special security groups (such as enterprise administrators). These two recipes help you to keep your AD free of stale objects or objects that could represent a security risk.

Systems used in this chapter

The recipes in this chapter, and the remainder of the book, are based on creating an AD domain, `Reskit.Org`, a child domain, `UK.Reskit.Org`, three domain controllers, and two additional servers. Here is a diagram showing the servers in use:

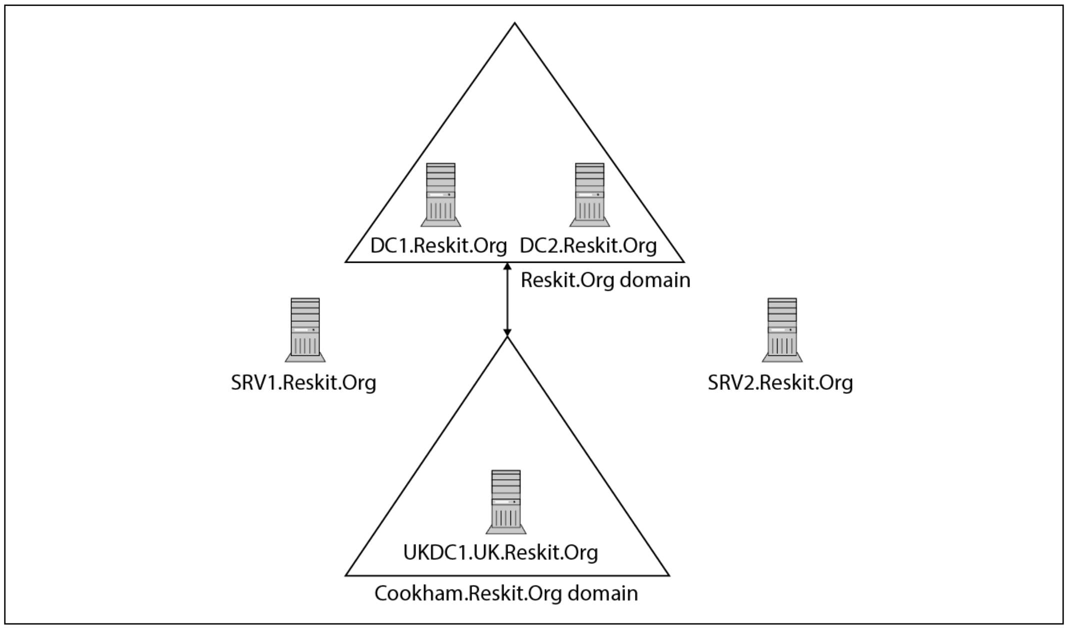


Figure 6.1: Reskit.Org forest diagram

You can use Hyper-V **virtual machines** (**VMs**) to implement this forest. To build these servers, you can get the build scripts from `https://github.com/doctordns/ReskitBuildScripts`. Each recipe in the chapter sets out the specific servers to use. You build new VMs as you need them.

Installing an AD forest root domain

You create an AD forest by creating your first domain controller. Installing Active Directory and DNS has always been reasonably straightforward. You can always use the Server Manager GUI, but using PowerShell is also straightforward.

To create a DC, you start with a system running Windows Server. You then add the AD DS Windows feature to the server. Finally, you create your first DC, for example, a single domain controller, `DC1.Reskit.Org`, for the `Reskit.Org` domain.

Getting ready

You run this recipe on `DC1`, a workgroup server on which you have installed PowerShell 7 and VS Code. You install this server as a workgroup server using the build scripts at `https://github.com/doctordns/ReskitBuildScripts`.

How to do it...

1. Installing the AD DS feature and management tools

```
Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
```

2. Importing the `ADDSDeployment` module

```
Import-Module -Name ADDSDeployment
```

3. Examining the commands in the `ADDSDeployment` module

```
Get-Command -Module ADDSDeployment
```

4. Creating a secure password for Administrator

```
$PSSHT = @{
  String      = 'Pa$$w0rd'
  AsPlainText = $true
  Force       = $true
}
$PSS = ConvertTo-SecureString @PSSHT
```

5. Testing DC forest installation starting on `DC1`

   ```
   $FOTHT = @{
     DomainName           = 'Reskit.Org'
     InstallDNS           = $true
     NoRebootOnCompletion = $true
     SafeModeAdministratorPassword = $PSS
     ForestMode           = 'WinThreshold'
     DomainMOde           = 'WinThreshold'
   }
   Test-ADDSForestInstallation @FOTHT -WarningAction SilentlyContinue
   ```

6. Creating forest root DC on `DC1`

   ```
   $ADHT = @{
     DomainName                    = 'Reskit.Org'
     SafeModeAdministratorPassword = $PSS
     InstallDNS                    = $true
     DomainMode                    = 'WinThreshold'
     ForestMode                    = 'WinThreshold'
     Force                         = $true
     NoRebootOnCompletion          = $true
     WarningAction                 = 'SilentlyContinue'
   }
   Install-ADDSForest @ADHT
   ```

7. Checking key AD and related services

   ```
   Get-Service -Name DNS, Netlogon
   ```

8. Checking DNS zones

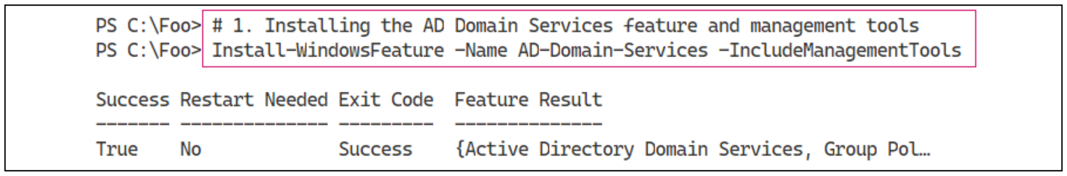
   ```
   Get-DnsServerZone
   ```

9. Restarting `DC1` to complete promotion

   ```
   Restart-Computer -Force
   ```

How it works...

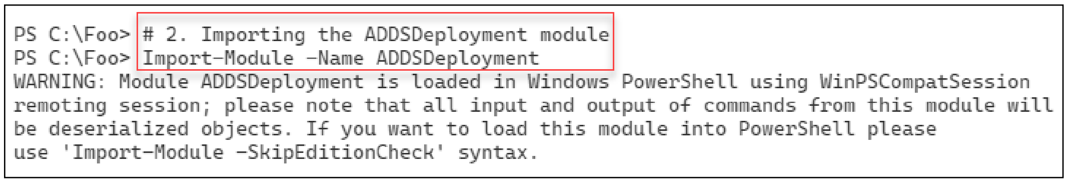
In *step 1*, you install the AD Domain Services feature. This feature enables you to deploy a server as a domain controller. The output of this command looks like this:



```
PS C:\Foo> # 1. Installing the AD Domain Services feature and management tools
PS C:\Foo> Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools

Success Restart Needed Exit Code  Feature Result
------- -------------- ---------  --------------
True    No             Success    {Active Directory Domain Services, Group Pol…
```

Figure 6.2: Installing the AD DS feature

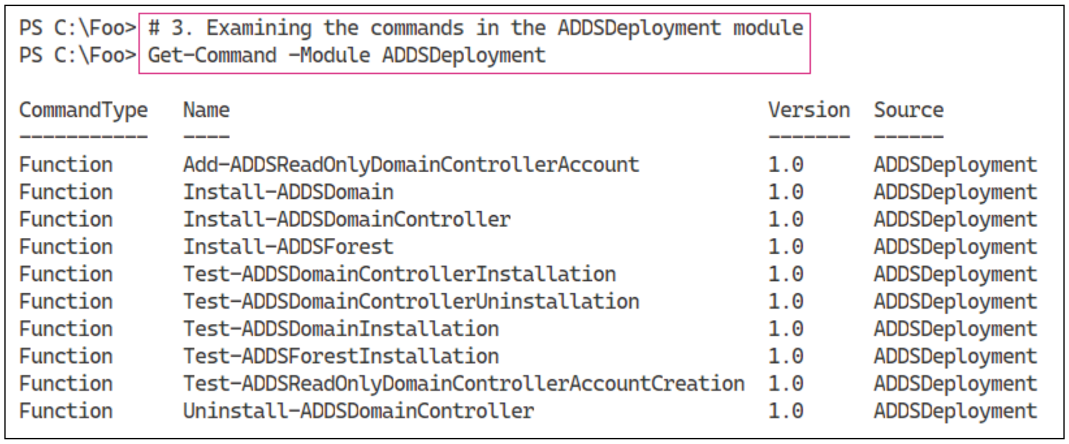
In *step 2*, you manually import the `ADDSDeployment` module. Since PowerShell does not support this module natively, this step loads the module using the Windows PowerShell Compatibility feature. The output of this command looks like this:



```
PS C:\Foo> # 2. Importing the ADDSDeployment module
PS C:\Foo> Import-Module -Name ADDSDeployment
WARNING: Module ADDSDeployment is loaded in Windows PowerShell using WinPSCompatSession
remoting session; please note that all input and output of commands from this module will
be deserialized objects. If you want to load this module into PowerShell please
use 'Import-Module -SkipEditionCheck' syntax.
```

Figure 6.3: Importing the ADDSDeployment module

In *step 3*, you use the `Get-Command` cmdlet to discover the commands contained in the `ADDSDeployment` module, which looks like this:

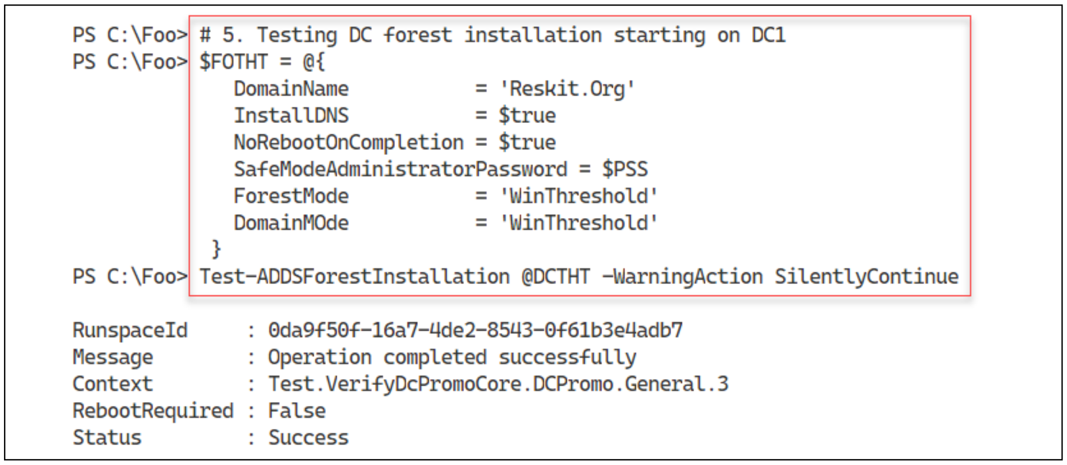


```
PS C:\Foo> # 3. Examining the commands in the ADDSDeployment module
PS C:\Foo> Get-Command -Module ADDSDeployment

CommandType     Name                                               Version    Source
-----------     ----                                               -------    ------
Function        Add-ADDSReadOnlyDomainControllerAccount            1.0        ADDSDeployment
Function        Install-ADDSDomain                                 1.0        ADDSDeployment
Function        Install-ADDSDomainController                       1.0        ADDSDeployment
Function        Install-ADDSForest                                 1.0        ADDSDeployment
Function        Test-ADDSDomainControllerInstallation              1.0        ADDSDeployment
Function        Test-ADDSDomainControllerUninstallation            1.0        ADDSDeployment
Function        Test-ADDSDomainInstallation                        1.0        ADDSDeployment
Function        Test-ADDSForestInstallation                        1.0        ADDSDeployment
Function        Test-ADDSReadOnlyDomainControllerAccountCreation   1.0        ADDSDeployment
Function        Uninstall-ADDSDomainController                     1.0        ADDSDeployment
```

Figure 6.4: Examining commands in the ADDSDeployment module

With *step 4*, you create a secure string password to use as the Administrator password in the domain you are creating. This step produces no output.

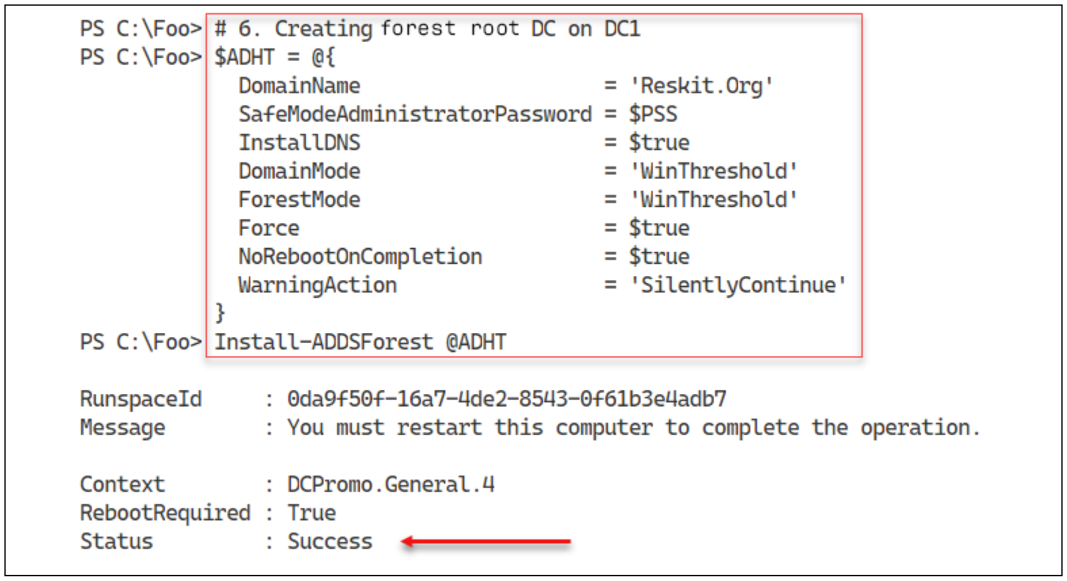
Before you promote a server to be a DC, it's useful to test to ensure that a promotion would be successful as far as possible. In *step 5*, you use the `Test-ADDSForestInstallation` command to check whether you can promote `DC1` as a DC in the `Reskit.Org` domain. The output of this command looks like this:



```
PS C:\Foo> # 5. Testing DC forest installation starting on DC1
PS C:\Foo> $FOTHT = @{
              DomainName           = 'Reskit.Org'
              InstallDNS           = $true
              NoRebootOnCompletion = $true
              SafeModeAdministratorPassword = $PSS
              ForestMode           = 'WinThreshold'
              DomainMOde           = 'WinThreshold'
            }
PS C:\Foo> Test-ADDSForestInstallation @DCTHT -WarningAction SilentlyContinue

RunspaceId     : 0da9f50f-16a7-4de2-8543-0f61b3e4adb7
Message        : Operation completed successfully
Context        : Test.VerifyDcPromoCore.DCPromo.General.3
RebootRequired : False
Status         : Success
```

Figure 6.5: Testing DC forest installation

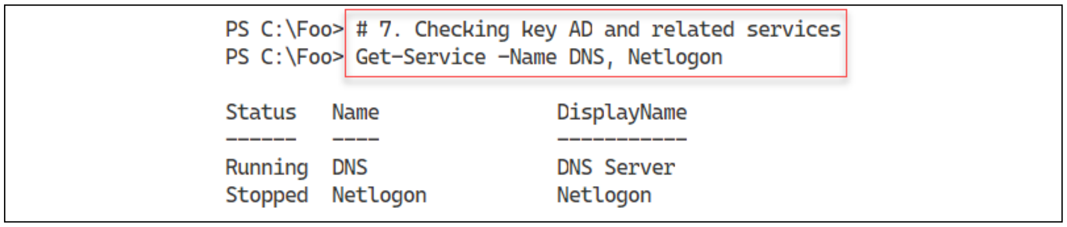
In *step 6*, you promote `DC1` to be the first domain controller in a new domain, `Reskit.Org`. The output looks like this:



```
PS C:\Foo> # 6. Creating forest root DC on DC1
PS C:\Foo> $ADHT = @{
             DomainName                    = 'Reskit.Org'
             SafeModeAdministratorPassword = $PSS
             InstallDNS                    = $true
             DomainMode                    = 'WinThreshold'
             ForestMode                    = 'WinThreshold'
             Force                         = $true
             NoRebootOnCompletion          = $true
             WarningAction                 = 'SilentlyContinue'
           }
PS C:\Foo> Install-ADDSForest @ADHT

RunspaceId     : 0da9f50f-16a7-4de2-8543-0f61b3e4adb7
Message        : You must restart this computer to complete the operation.

Context        : DCPromo.General.4
RebootRequired : True
Status         : Success
```

Figure 6.6: Creating a forest root DC on DC1

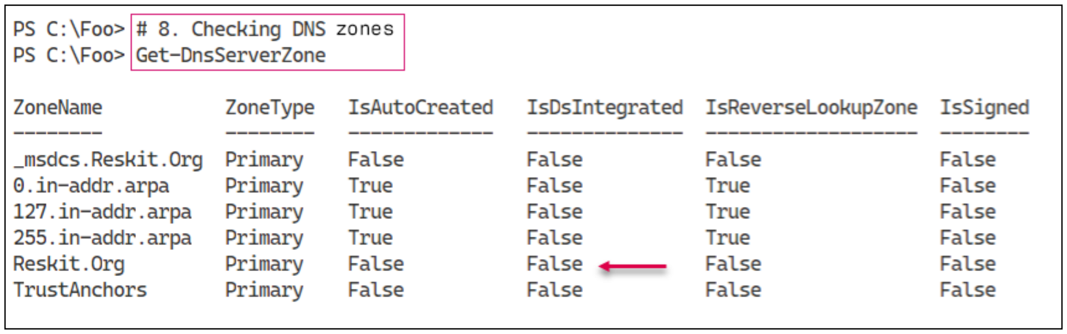
After the promotion is complete, you can check the critical services required for Active Directory. Checking the Netlogon and DNS services, which you do in *step 7*, should look like this:



```
PS C:\Foo> # 7. Checking key AD and related services
PS C:\Foo> Get-Service -Name DNS, Netlogon

Status   Name               DisplayName
------   ----               -----------
Running  DNS                DNS Server
Stopped  Netlogon           Netlogon
```

Figure 6.7: Checking the Netlogon and DNS services

When you promoted `DC1`, you also requested the promotion to install DNS on `DC1`. In *step 8*, you check on the zones created by the DC promotion process, which looks like this:



```
PS C:\Foo> # 8. Checking DNS zones
PS C:\Foo> Get-DnsServerZone

ZoneName            ZoneType  IsAutoCreated  IsDsIntegrated  IsReverseLookupZone  IsSigned
--------            --------  -------------  --------------  -------------------  --------
_msdcs.Reskit.Org   Primary   False          False           False                False
0.in-addr.arpa      Primary   True           False           True                 False
127.in-addr.arpa    Primary   True           False           True                 False
255.in-addr.arpa    Primary   True           False           True                 False
Reskit.Org          Primary   False          False           False                False
TrustAnchors        Primary   False          False           False                False
```

Figure 6.8: Checking DNS zones

You need to reboot DC1 to complete the promotion process, which you do in *step 9*, generating no actual output.

There's more...

The cmdlets that enable you to promote a server to be a DC are not installed on a server system by default. Adding the Active Directory Domain Services Windows feature, in *step 1*, adds the necessary cmdlets to the system.

In *step 6*, you install AD and direct that a DNS server should also be installed – and you check for its presence in *step 7*. In *step 8*, you view the DNS zones created automatically by the promotion process. You specify the DNS domain name, `Reskit.Org`, using the `DomainName` parameter to `Install-ADDSForest`. This step creates the DNS domain, but it is, at this point, still a *non*-AD-integrated zone. Once you reboot the service, this zone should become AD-integrated (and set for secure updates only).

Once you complete the verification of a successful AD installation, you reboot the server. After the restart, there are further tests that you should run, as we show in the next recipe, *Testing an AD installation*.

Testing an AD installation

In *Installing an AD forest root domain*, you installed AD on `DC1`. In that recipe, you installed AD (without rebooting) and tested certain services. After the required reboot (which you completed at the end of the previous recipe), it is useful to check to ensure that your domain is fully up and running and working correctly. In this recipe, you examine core aspects of the AD infrastructure on your first DC.

Getting ready

You run this recipe on `DC1`, the first domain controller in the `Reskit.Org` domain, after you have promoted it to be a DC. You created `DC1` as a domain controller in the Reskit.Org domain in *Installing an AD forest root domain*. Log on as `Reskit\Administrator`.

How to do it...

1. Examining AD root **directory service entry** (**DSE**)

   ```
   Get-ADRootDSE -Server DC1.Reskit.Org
   ```

2. Viewing AD forest details

   ```
   Get-ADForest
   ```

3. Viewing AD domain details

```
Get-ADDomain
```

4. Checking Netlogon, ADWS, and DNS services

```
Get-Service NetLogon, ADWS, DNS
```

5. Getting initial AD users

```
Get-ADUser -Filter * |
  Sort-Object -Property Name |
    Format-Table -Property Name, DistinguishedName
```

6. Getting initial AD groups

```
Get-ADGroup -Filter *  |
  Sort-Object -Property Groupscope, Name |
    Format-Table -Property Name, GroupScope
```

7. Examining Enterprise Admins group membership

```
Get-ADGroupMember -Identity 'Enterprise Admins'
```

8. Checking DNS zones on `DC1`

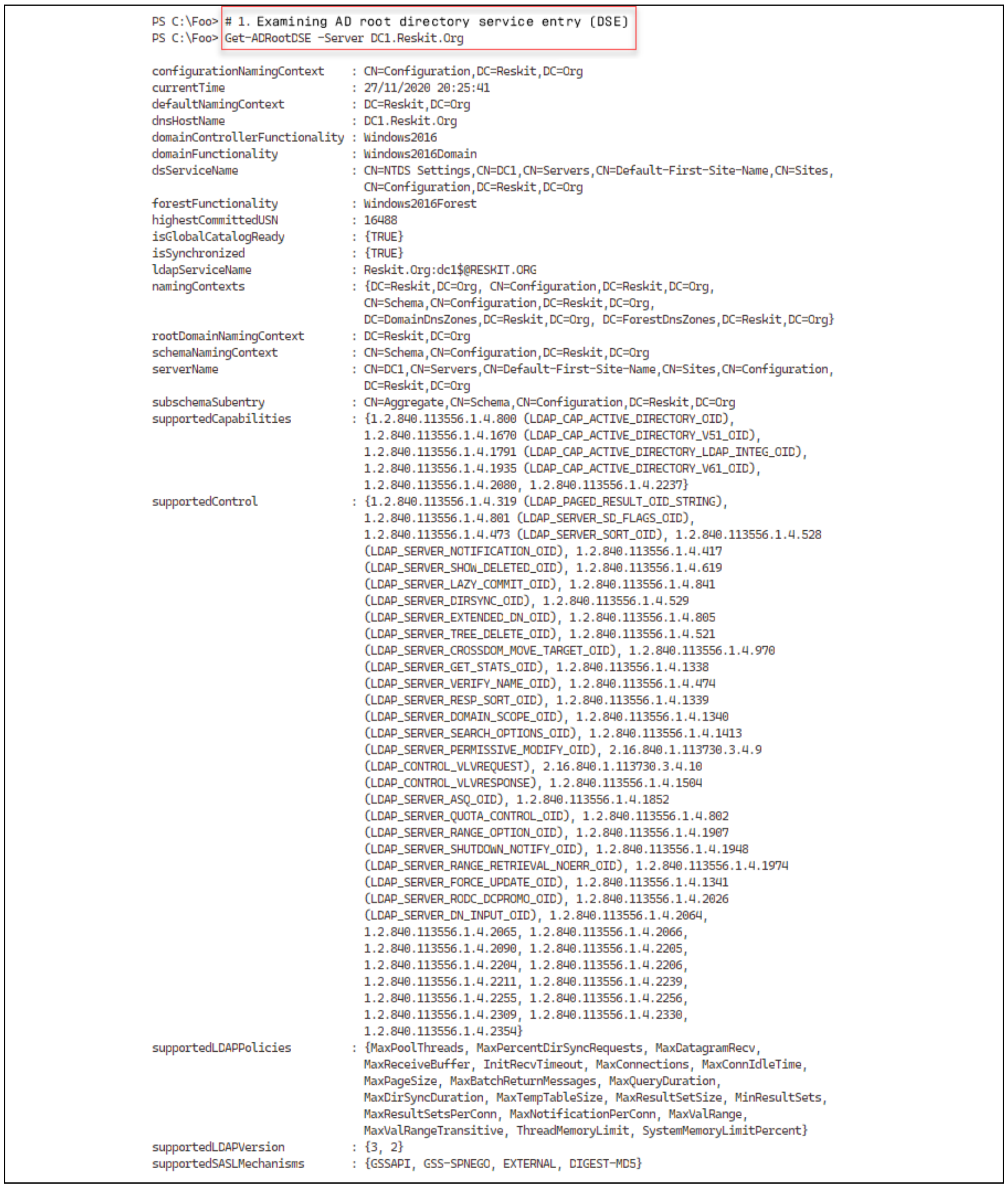
```
Get-DnsServerZone -ComputerName DC1
```

9. Testing domain name DNS resolution

```
Resolve-DnsName -Name Reskit.Org
```

How it works...

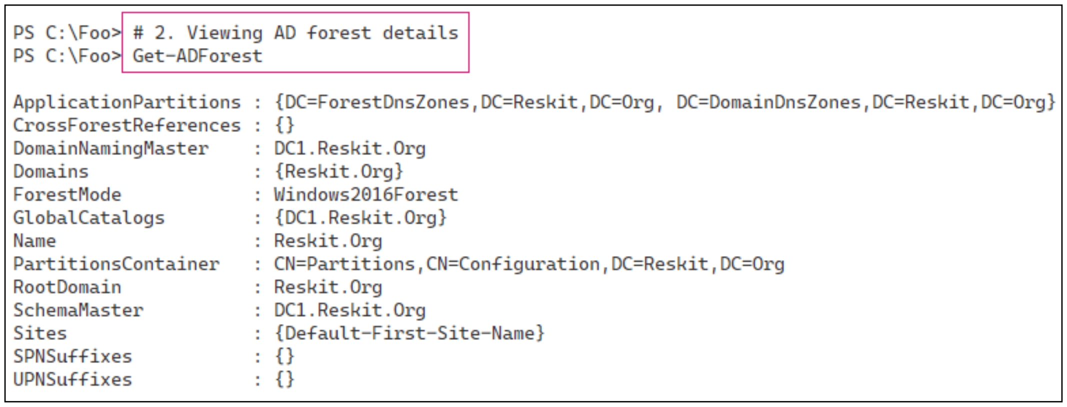
After you've completed the installation of AD and rebooted, in *step 1*, you examine the AD DSE, which looks like this:



```
PS C:\Foo> # 1. Examining AD root directory service entry (DSE)
PS C:\Foo> Get-ADRootDSE -Server DC1.Reskit.Org

configurationNamingContext    : CN=Configuration,DC=Reskit,DC=Org
currentTime                   : 27/11/2020 20:25:41
defaultNamingContext          : DC=Reskit,DC=Org
dnsHostName                   : DC1.Reskit.Org
domainControllerFunctionality : Windows2016
domainFunctionality           : Windows2016Domain
dsServiceName                 : CN=NTDS Settings,CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,
                                CN=Configuration,DC=Reskit,DC=Org
forestFunctionality           : Windows2016Forest
highestCommittedUSN           : 16488
isGlobalCatalogReady          : {TRUE}
isSynchronized                : {TRUE}
ldapServiceName               : Reskit.Org:dc1$@RESKIT.ORG
namingContexts                : {DC=Reskit,DC=Org, CN=Configuration,DC=Reskit,DC=Org,
                                CN=Schema,CN=Configuration,DC=Reskit,DC=Org,
                                DC=DomainDnsZones,DC=Reskit,DC=Org, DC=ForestDnsZones,DC=Reskit,DC=Org}
rootDomainNamingContext       : DC=Reskit,DC=Org
schemaNamingContext           : CN=Schema,CN=Configuration,DC=Reskit,DC=Org
serverName                    : CN=DC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,
                                DC=Reskit,DC=Org
subschemaSubentry             : CN=Aggregate,CN=Schema,CN=Configuration,DC=Reskit,DC=Org
supportedCapabilities         : {1.2.840.113556.1.4.800 (LDAP_CAP_ACTIVE_DIRECTORY_OID),
                                1.2.840.113556.1.4.1670 (LDAP_CAP_ACTIVE_DIRECTORY_V51_OID),
                                1.2.840.113556.1.4.1791 (LDAP_CAP_ACTIVE_DIRECTORY_LDAP_INTEG_OID),
                                1.2.840.113556.1.4.1935 (LDAP_CAP_ACTIVE_DIRECTORY_V61_OID),
                                1.2.840.113556.1.4.2080, 1.2.840.113556.1.4.2237}
supportedControl              : {1.2.840.113556.1.4.319 (LDAP_PAGED_RESULT_OID_STRING),
                                1.2.840.113556.1.4.801 (LDAP_SERVER_SD_FLAGS_OID),
                                1.2.840.113556.1.4.473 (LDAP_SERVER_SORT_OID), 1.2.840.113556.1.4.528
                                (LDAP_SERVER_NOTIFICATION_OID), 1.2.840.113556.1.4.417
                                (LDAP_SERVER_SHOW_DELETED_OID), 1.2.840.113556.1.4.619
                                (LDAP_SERVER_LAZY_COMMIT_OID), 1.2.840.113556.1.4.841
                                (LDAP_SERVER_DIRSYNC_OID), 1.2.840.113556.1.4.529
                                (LDAP_SERVER_EXTENDED_DN_OID), 1.2.840.113556.1.4.805
                                (LDAP_SERVER_TREE_DELETE_OID), 1.2.840.113556.1.4.521
                                (LDAP_SERVER_CROSSDOM_MOVE_TARGET_OID), 1.2.840.113556.1.4.970
                                (LDAP_SERVER_GET_STATS_OID), 1.2.840.113556.1.4.1338
                                (LDAP_SERVER_VERIFY_NAME_OID), 1.2.840.113556.1.4.474
                                (LDAP_SERVER_RESP_SORT_OID), 1.2.840.113556.1.4.1339
                                (LDAP_SERVER_DOMAIN_SCOPE_OID), 1.2.840.113556.1.4.1340
                                (LDAP_SERVER_SEARCH_OPTIONS_OID), 1.2.840.113556.1.4.1413
                                (LDAP_SERVER_PERMISSIVE_MODIFY_OID), 2.16.840.1.113730.3.4.9
                                (LDAP_CONTROL_VLVREQUEST), 2.16.840.1.113730.3.4.10
                                (LDAP_CONTROL_VLVRESPONSE), 1.2.840.113556.1.4.1504
                                (LDAP_SERVER_ASQ_OID), 1.2.840.113556.1.4.1852
                                (LDAP_SERVER_QUOTA_CONTROL_OID), 1.2.840.113556.1.4.802
                                (LDAP_SERVER_RANGE_OPTION_OID), 1.2.840.113556.1.4.1907
                                (LDAP_SERVER_SHUTDOWN_NOTIFY_OID), 1.2.840.113556.1.4.1948
                                (LDAP_SERVER_RANGE_RETRIEVAL_NOERR_OID), 1.2.840.113556.1.4.1974
                                (LDAP_SERVER_FORCE_UPDATE_OID), 1.2.840.113556.1.4.1341
                                (LDAP_SERVER_RODC_DCPROMO_OID), 1.2.840.113556.1.4.2026
                                (LDAP_SERVER_DN_INPUT_OID), 1.2.840.113556.1.4.2064,
                                1.2.840.113556.1.4.2065, 1.2.840.113556.1.4.2066,
                                1.2.840.113556.1.4.2090, 1.2.840.113556.1.4.2205,
                                1.2.840.113556.1.4.2204, 1.2.840.113556.1.4.2206,
                                1.2.840.113556.1.4.2211, 1.2.840.113556.1.4.2239,
                                1.2.840.113556.1.4.2255, 1.2.840.113556.1.4.2256,
                                1.2.840.113556.1.4.2309, 1.2.840.113556.1.4.2330,
                                1.2.840.113556.1.4.2354}
supportedLDAPPolicies         : {MaxPoolThreads, MaxPercentDirSyncRequests, MaxDatagramRecv,
                                MaxReceiveBuffer, InitRecvTimeout, MaxConnections, MaxConnIdleTime,
                                MaxPageSize, MaxBatchReturnMessages, MaxQueryDuration,
                                MaxDirSyncDuration, MaxTempTableSize, MaxResultSetSize, MinResultSets,
                                MaxResultSetsPerConn, MaxNotificationPerConn, MaxValRange,
                                MaxValRangeTransitive, ThreadMemoryLimit, SystemMemoryLimitPercent}
supportedLDAPVersion          : {3, 2}
supportedSASLMechanisms       : {GSSAPI, GSS-SPNEGO, EXTERNAL, DIGEST-MD5}
```

Figure 6.9: Examining the AD DSE

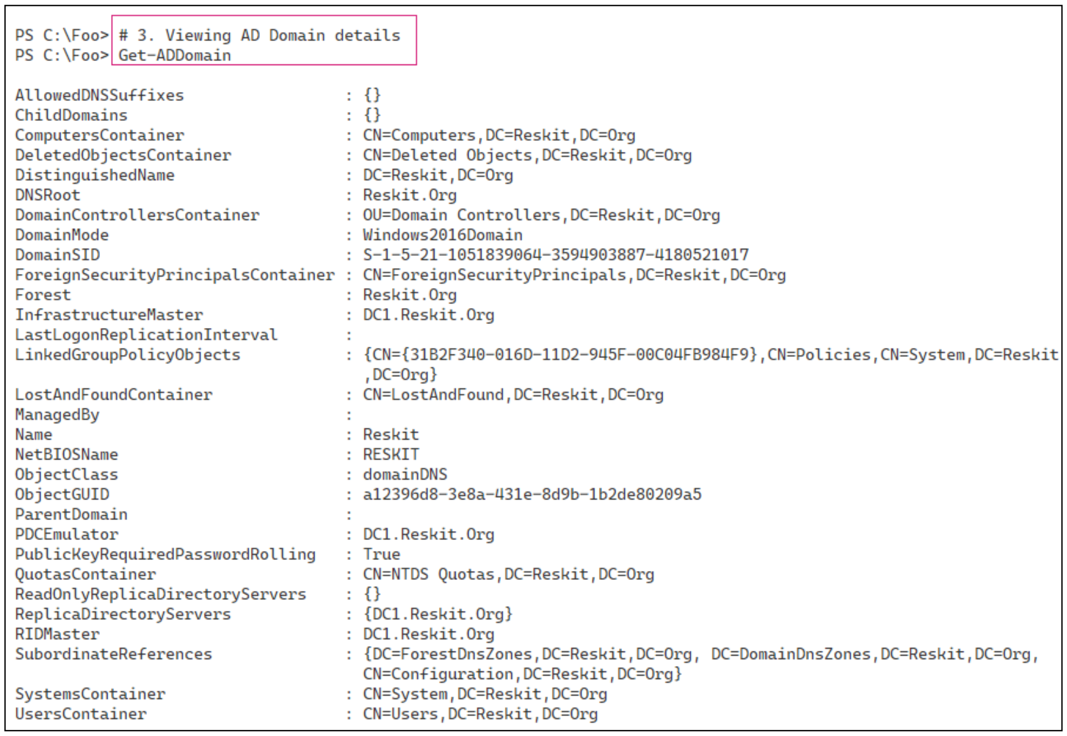
In *step 2*, you use the `Get-ADForest` command to review further information on the newly created forest, which looks like this:



```
PS C:\Foo> # 2. Viewing AD forest details
PS C:\Foo> Get-ADForest

ApplicationPartitions : {DC=ForestDnsZones,DC=Reskit,DC=Org, DC=DomainDnsZones,DC=Reskit,DC=Org}
CrossForestReferences : {}
DomainNamingMaster    : DC1.Reskit.Org
Domains               : {Reskit.Org}
ForestMode            : Windows2016Forest
GlobalCatalogs        : {DC1.Reskit.Org}
Name                  : Reskit.Org
PartitionsContainer   : CN=Partitions,CN=Configuration,DC=Reskit,DC=Org
RootDomain            : Reskit.Org
SchemaMaster          : DC1.Reskit.Org
Sites                 : {Default-First-Site-Name}
SPNSuffixes           : {}
UPNSuffixes           : {}
```

Figure 6.10: Viewing the AD forest details

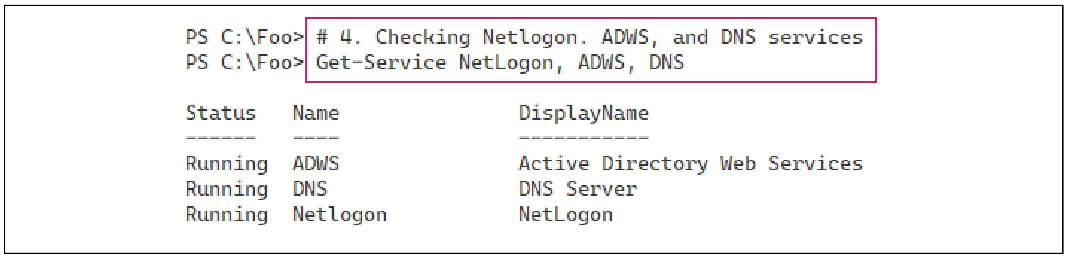
In *step 3*, you use `Get-ADDomain` to return more information about the `Reskit.Org` domain, which looks like this:



```
PS C:\Foo> # 3. Viewing AD Domain details
PS C:\Foo> Get-ADDomain

AllowedDNSSuffixes                 : {}
ChildDomains                       : {}
ComputersContainer                 : CN=Computers,DC=Reskit,DC=Org
DeletedObjectsContainer            : CN=Deleted Objects,DC=Reskit,DC=Org
DistinguishedName                  : DC=Reskit,DC=Org
DNSRoot                            : Reskit.Org
DomainControllersContainer         : OU=Domain Controllers,DC=Reskit,DC=Org
DomainMode                         : Windows2016Domain
DomainSID                          : S-1-5-21-1051839064-3594903887-4180521017
ForeignSecurityPrincipalsContainer : CN=ForeignSecurityPrincipals,DC=Reskit,DC=Org
Forest                             : Reskit.Org
InfrastructureMaster               : DC1.Reskit.Org
LastLogonReplicationInterval       :
LinkedGroupPolicyObjects           : {CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=Reskit
                                     ,DC=Org}
LostAndFoundContainer              : CN=LostAndFound,DC=Reskit,DC=Org
ManagedBy                          :
Name                               : Reskit
NetBIOSName                        : RESKIT
ObjectClass                        : domainDNS
ObjectGUID                         : a12396d8-3e8a-431e-8d9b-1b2de80209a5
ParentDomain                       :
PDCEmulator                        : DC1.Reskit.Org
PublicKeyRequiredPasswordRolling   : True
QuotasContainer                    : CN=NTDS Quotas,DC=Reskit,DC=Org
ReadOnlyReplicaDirectoryServers    : {}
ReplicaDirectoryServers            : {DC1.Reskit.Org}
RIDMaster                          : DC1.Reskit.Org
SubordinateReferences              : {DC=ForestDnsZones,DC=Reskit,DC=Org, DC=DomainDnsZones,DC=Reskit,DC=Org,
                                     CN=Configuration,DC=Reskit,DC=Org}
SystemsContainer                   : CN=System,DC=Reskit,DC=Org
UsersContainer                     : CN=Users,DC=Reskit,DC=Org
```

Figure 6.11: Viewing AD domain details

The AD services run within the Netlogon Windows service. The ADWS service is a web service used by the PowerShell AD cmdlets to communicate with the AD. AD relies on DNS as a locator service, enabling AD clients and DCs to find DCs. All three must be up and running for you to manage AD using PowerShell. In *step 4*, you check all three services, which looks like this:

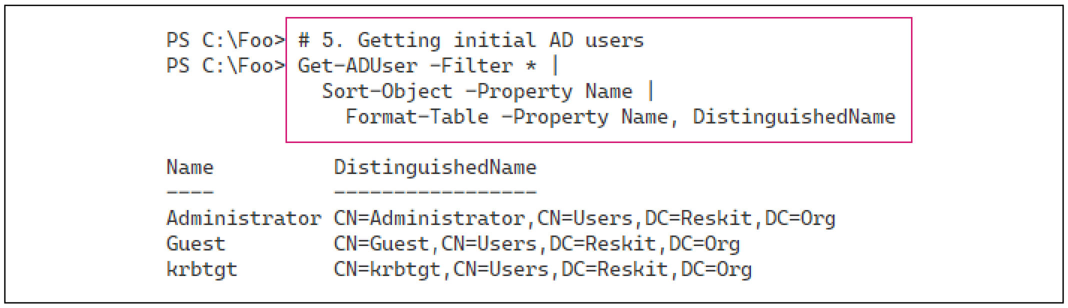


```
PS C:\Foo> # 4. Checking Netlogon. ADWS, and DNS services
PS C:\Foo> Get-Service NetLogon, ADWS, DNS

Status   Name               DisplayName
------   ----               -----------
Running  ADWS               Active Directory Web Services
Running  DNS                DNS Server
Running  Netlogon           NetLogon
```

Figure 6.12: Checking the Netlogon, ADWS, and DNS services

In *Installing an AD forest root domain*, you promoted `DC1`, a Windows server, to be a DC. In doing so, the promotion process creates three domain users: Administrator, Guest, and the krbtgt (Kerberos Ticket-Granting Ticket) user. You should never touch the krbtgt user and should usually leave the Guest user disabled. For added security, you can rename the Administrator user to something less easy to guess and create a new, very low-privilege user with the name Administrator.

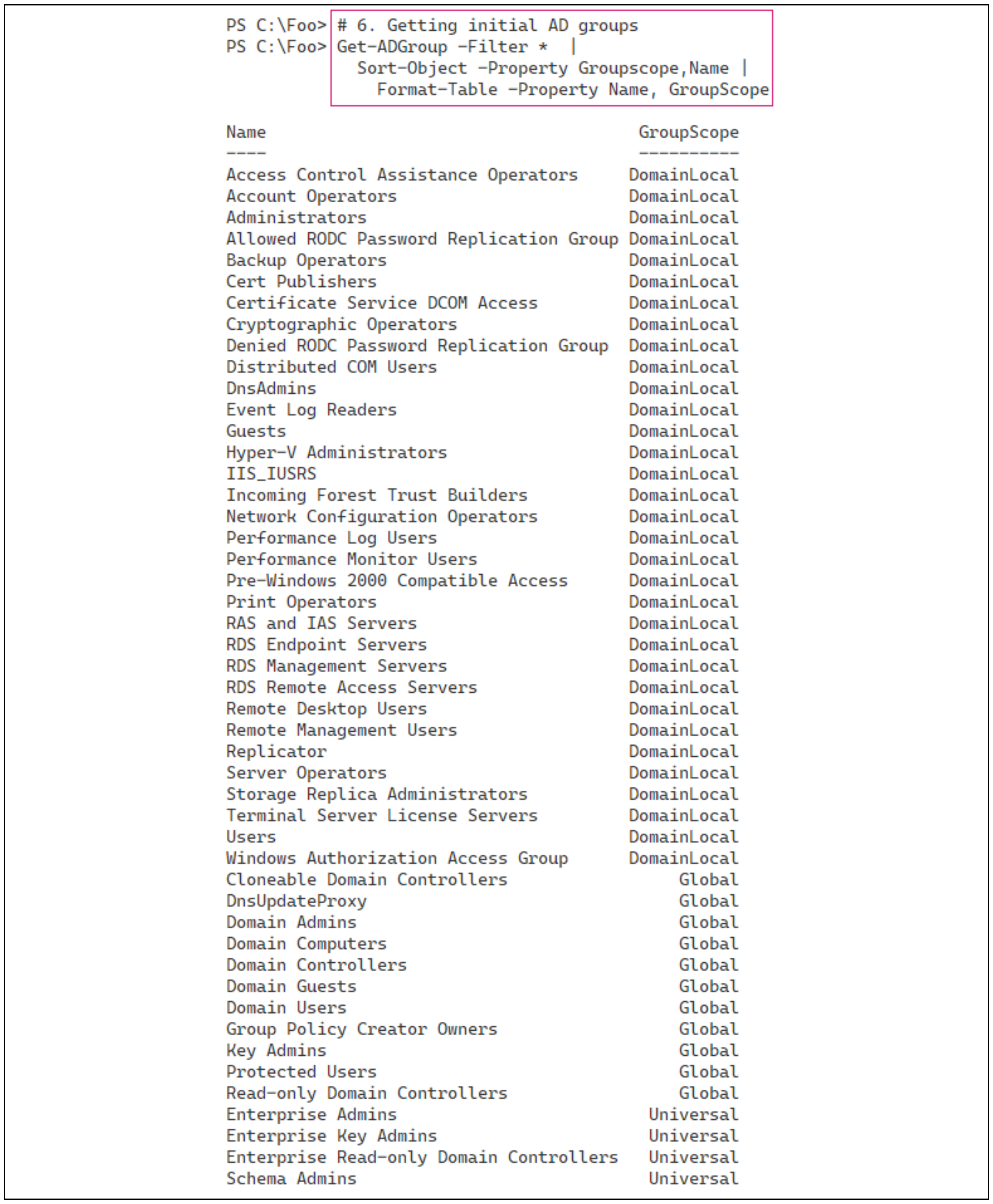
In *step 5*, you examine the users in your newly created AD forest, which looks like this:



```
PS C:\Foo> # 5. Getting initial AD users
PS C:\Foo> Get-ADUser -Filter * |
             Sort-Object -Property Name |
               Format-Table -Property Name, DistinguishedName

Name          DistinguishedName
----          -----------------
Administrator CN=Administrator,CN=Users,DC=Reskit,DC=Org
Guest         CN=Guest,CN=Users,DC=Reskit,DC=Org
krbtgt        CN=krbtgt,CN=Users,DC=Reskit,DC=Org
```

Figure 6.13: Getting initial AD users in the new AD forest

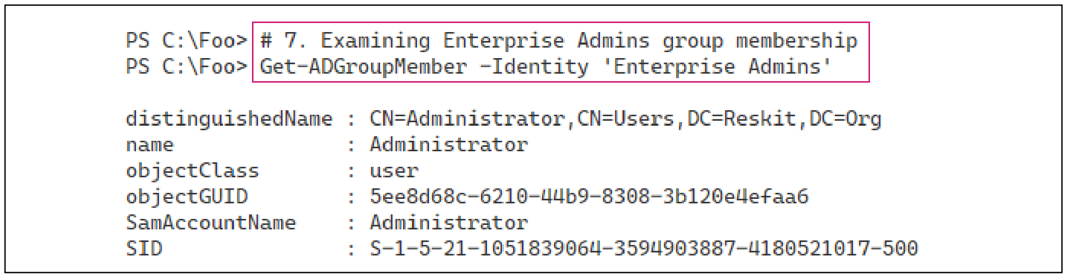
The DC promotion process also creates several groups that can be useful. In *step 6*, you examine the different groups created (and their scope), which looks like this:



```
PS C:\Foo> # 6. Getting initial AD groups
PS C:\Foo> Get-ADGroup -Filter *  |
             Sort-Object -Property Groupscope,Name |
               Format-Table -Property Name, GroupScope

Name                                    GroupScope
----                                    ----------
Access Control Assistance Operators    DomainLocal
Account Operators                      DomainLocal
Administrators                         DomainLocal
Allowed RODC Password Replication Group DomainLocal
Backup Operators                       DomainLocal
Cert Publishers                        DomainLocal
Certificate Service DCOM Access        DomainLocal
Cryptographic Operators                DomainLocal
Denied RODC Password Replication Group  DomainLocal
Distributed COM Users                  DomainLocal
DnsAdmins                              DomainLocal
Event Log Readers                      DomainLocal
Guests                                 DomainLocal
Hyper-V Administrators                 DomainLocal
IIS_IUSRS                              DomainLocal
Incoming Forest Trust Builders         DomainLocal
Network Configuration Operators        DomainLocal
Performance Log Users                  DomainLocal
Performance Monitor Users              DomainLocal
Pre-Windows 2000 Compatible Access     DomainLocal
Print Operators                        DomainLocal
RAS and IAS Servers                    DomainLocal
RDS Endpoint Servers                   DomainLocal
RDS Management Servers                 DomainLocal
RDS Remote Access Servers              DomainLocal
Remote Desktop Users                   DomainLocal
Remote Management Users                DomainLocal
Replicator                             DomainLocal
Server Operators                       DomainLocal
Storage Replica Administrators         DomainLocal
Terminal Server License Servers        DomainLocal
Users                                  DomainLocal
Windows Authorization Access Group     DomainLocal
Cloneable Domain Controllers                Global
DnsUpdateProxy                              Global
Domain Admins                               Global
Domain Computers                            Global
Domain Controllers                          Global
Domain Guests                               Global
Domain Users                                Global
Group Policy Creator Owners                 Global
Key Admins                                  Global
Protected Users                             Global
Read-only Domain Controllers                Global
Enterprise Admins                        Universal
Enterprise Key Admins                    Universal
Enterprise Read-only Domain Controllers  Universal
Schema Admins                            Universal
```

Figure 6.14: Examining initial AD groups and their scope

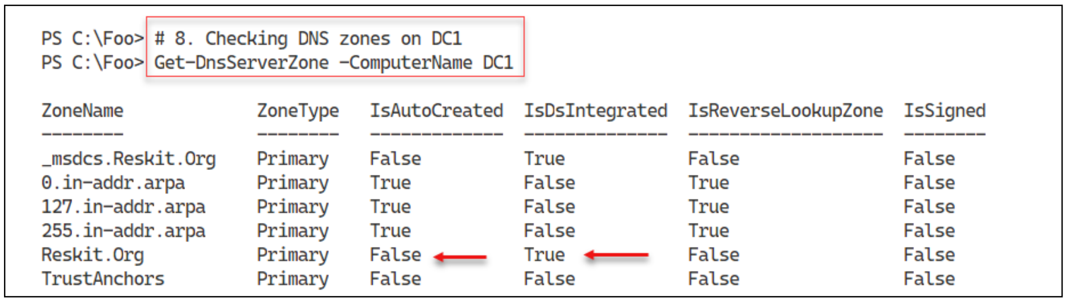
The Enterprise Admins group is one with very high privilege. Members of this group can perform just about any operation across the domain – stop/start services, modify the AD, and access any file or folder on the domain or any domain-joined system. In *step 7*, you examine the initial members of this high-privilege group, which looks like this:



```
PS C:\Foo> # 7. Examining Enterprise Admins group membership
PS C:\Foo> Get-ADGroupMember -Identity 'Enterprise Admins'

distinguishedName : CN=Administrator,CN=Users,DC=Reskit,DC=Org
name              : Administrator
objectClass       : user
objectGUID        : 5ee8d68c-6210-44b9-8308-3b120e4efaa6
SamAccountName    : Administrator
SID               : S-1-5-21-1051839064-3594903887-4180521017-500
```

Figure 6.15: Examining Enterprise Admins group membership

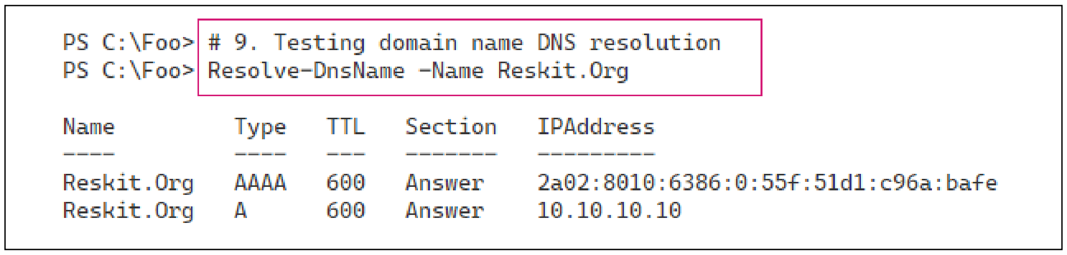
AD relies on DNS to enable an AD client or AD DC to find DCs. When you install a DC, you can also install the DNS service at the same time. When you install a DC with DNS, the AD promotion process creates several DNS zones in your newly created DNS service. In *step 8*, you examine the DNS zones created on `DC1`, which looks like this:



```
PS C:\Foo> # 8. Checking DNS zones on DC1
PS C:\Foo> Get-DnsServerZone -ComputerName DC1

ZoneName            ZoneType  IsAutoCreated  IsDsIntegrated  IsReverseLookupZone  IsSigned
--------            --------  -------------  --------------  -------------------  --------
_msdcs.Reskit.Org   Primary   False          True            False                False
0.in-addr.arpa      Primary   True           False           True                 False
127.in-addr.arpa    Primary   True           False           True                 False
255.in-addr.arpa    Primary   True           False           True                 False
Reskit.Org          Primary   False          True            False                False
TrustAnchors        Primary   False          False           False                False
```

Figure 6.16: Checking the DNS zones created on DC1

Another good test of your newly promoted DC is to ensure you can resolve the DNS name of your new domain (`Reskit.Org`). In *step 9*, you use the `Resolve-DnsName` to check, which looks like this:



```
PS C:\Foo> # 9. Testing domain name DNS resolution
PS C:\Foo> Resolve-DnsName -Name Reskit.Org

Name        Type  TTL  Section  IPAddress
----        ----  ---  -------  ---------
Reskit.Org  AAAA  600  Answer   2a02:8010:6386:0:55f:51d1:c96a:bafe
Reskit.Org  A     600  Answer   10.10.10.10
```

Figure 6.17: Testing domain name DNS resolution

There's more...

In *step 1*, you viewed the DSE for your domain. The AD implements a **Lightweight Directory Access Protocol** (**LDAP**) directory service for domain activities. The DSE is a component of LDAP directories and contains information about your directory structure. The DSE is available without requiring authentication and contains much information about your AD forest. That information could help an attacker; thus, best practice is never to expose an AD DC to the Internet if you can help it. For a more detailed look at the root DSE, see `https://docs.microsoft.com/windows/win32/adschema/rootdse`.

The ADWS service, which you investigate in *step 4*, implements a web service. The AD commands use this web service to get information from and make changes to your AD. If this service is not running, AD commands do not work. You should always check to ensure the service has started before proceeding to use the AD cmdlets.

In *step 6*, you saw the groups created by the promotion process. These groups have permissions associated and thus are useful. Before adding users to these groups, consider reviewing the group and determining (and possibly changing) the permissions and rights assigned to these groups.

Installing a replica domain controller

In *Installing an AD forest root domain*, you installed AD on `DC1`. If you have just one DC, then that DC is a single point of failure. With a single DC, when `DC1` goes down, you cannot manage AD using the PowerShell cmdlets, and users cannot log on. It is always best practice to install at least two DCs. If you are using VMs for your DCs, you should also ensure that each DC VM is on a separate virtualization host.

To add a second DC to your domain, you run `Install-ADDSDomainController` on another host, for example, `DC2`. This cmdlet is similar to `Install-ADDSForest` in terms of parameters. As with creating your first DC, it is useful to carry out some tests to ensure the second DC's promotion can succeed.

In this recipe, you promote a host, `DC2`, to be the second DC in the `Reskit.Org` domain. Like when creating your first DC, after you promote `DC2` to be a DC, you need to reboot the server before processing. And after the reboot, it is useful to ensure the promotion process was successful.

Getting ready

You run this recipe on `DC2`, a domain-joined server on which you have installed PowerShell 7 and VS Code. You should log into `DC2` as `Reskit\Administrator`, a member of the Enterprise Admins group.

Using the `Reskit.Org` build scripts, you would build this VM after creating `DC1` to be a DC.

How to do it...

1. Importing the `ServerManager` module

```
Import-Module -Name ServerManager -WarningAction SilentlyContinue
```

2. Checking `DC1` can be resolved

```
Resolve-DnsName -Name DC1.Reskit.Org -Type A
```

3. Testing the network connection to `DC1`

```
Test-NetConnection -ComputerName DC1.Reskit.Org -Port 445
Test-NetConnection -ComputerName DC1.Reskit.Org -Port 389
```

4. Adding the AD DS features on `DC2`

```
Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
```

5. Promoting `DC2` to be a DC

```
Import-Module -Name ADDSDeployment -WarningAction SilentlyContinue
$URK    = 'Administrator@Reskit.Org'
$PW     = 'Pa$$w0rd'
$PSS    = ConvertTo-SecureString -String $PW -AsPlainText -Force
$CredRK = [PSCredential]::New($URK,$PSS)
$INSTALLHT = @{
  DomainName                    = 'Reskit.Org'
  SafeModeAdministratorPassword = $PSS
  SiteName                      = 'Default-First-Site-Name'
  NoRebootOnCompletion          = $true
  InstallDNS                    = $false
  Credential                    = $CredRK
  Force                         = $true
  }
Install-ADDSDomainController @INSTALLHT | Out-Null
```

6. Checking the computer objects in AD

```
Get-ADComputer -Filter * |
  Format-Table DNSHostName, DistinguishedName
```

7. Rebooting `DC2` manually

```
Restart-Computer -Force
```

8. Checking DCs in `Reskit.Org`

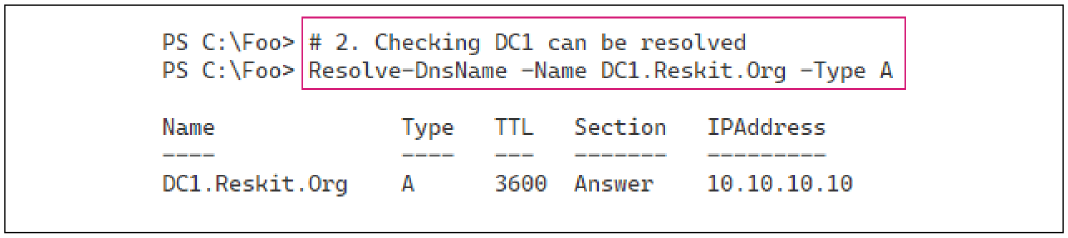
```
$SB = 'OU=Domain Controllers,DC=Reskit,DC=Org'
Get-ADComputer -Filter * -SearchBase $SB |
  Format-Table -Property DNSHostName, Enabled
```

9. Viewing `Reskit.Org` domain DCs

```
Get-ADDomain |
  Format-Table -Property Forest, Name, Replica*
```

How it works...

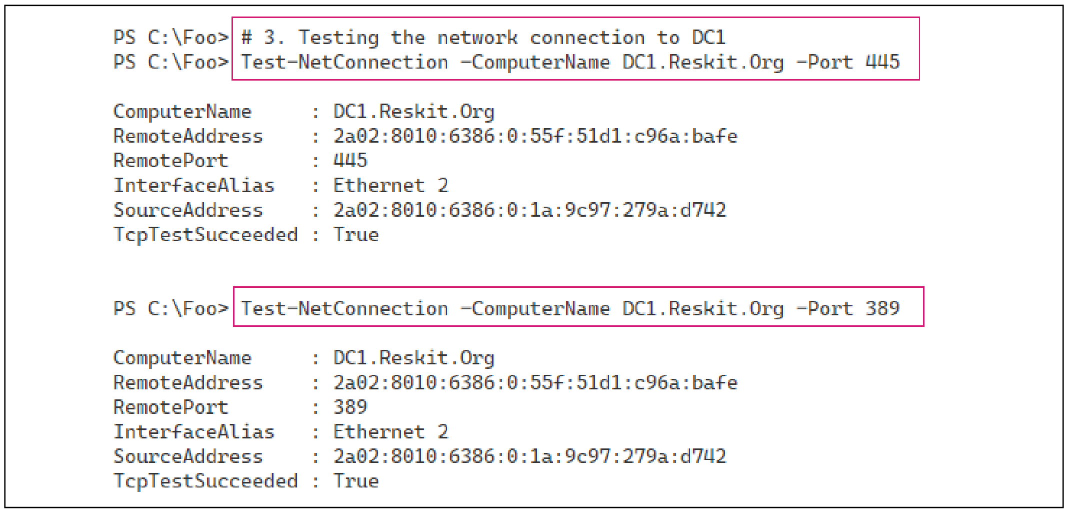
In *step 1*, you import the `ServerManager` module, which creates no output. With *step 2*, you ensure that you can resolve your DC's address, which is, at this point, the only DC in the `Reskit.Org` domain. The output looks like this:



```
PS C:\Foo> # 2. Checking DC1 can be resolved
PS C:\Foo> Resolve-DnsName -Name DC1.Reskit.Org -Type A

Name                 Type   TTL   Section    IPAddress
----                 ----   ---   -------    ---------
DC1.Reskit.Org       A      3600  Answer     10.10.10.10
```

Figure 6.18: Checking DC1 can be resolved

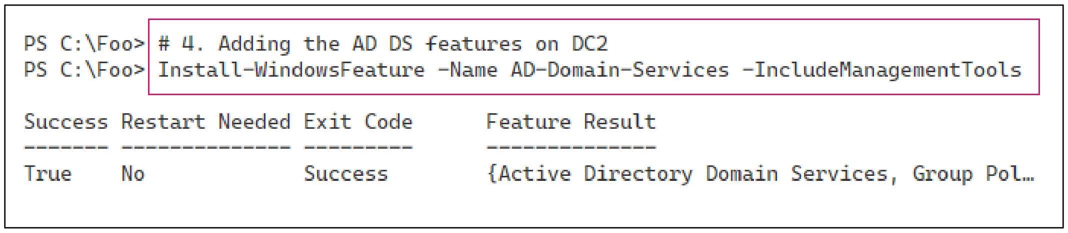
After confirming that you can resolve the IP address of DC1, in *step 3*, you check that you can connect to two key ports on `DC1` (`445` and `389`). The output looks like this:



```
PS C:\Foo> # 3. Testing the network connection to DC1
PS C:\Foo> Test-NetConnection -ComputerName DC1.Reskit.Org -Port 445

ComputerName     : DC1.Reskit.Org
RemoteAddress    : 2a02:8010:6386:0:55f:51d1:c96a:bafe
RemotePort       : 445
InterfaceAlias   : Ethernet 2
SourceAddress    : 2a02:8010:6386:0:1a:9c97:279a:d742
TcpTestSucceeded : True


PS C:\Foo> Test-NetConnection -ComputerName DC1.Reskit.Org -Port 389

ComputerName     : DC1.Reskit.Org
RemoteAddress    : 2a02:8010:6386:0:55f:51d1:c96a:bafe
RemotePort       : 389
InterfaceAlias   : Ethernet 2
SourceAddress    : 2a02:8010:6386:0:1a:9c97:279a:d742
TcpTestSucceeded : True
```

Figure 6.19: Testing the network connection to DC1

As you saw, when promoting `DC1` to be your first DC, you need to add the `ADDSDeployment` feature to `DC2` before you can promote the DC. The output from *step 4* looks like this:

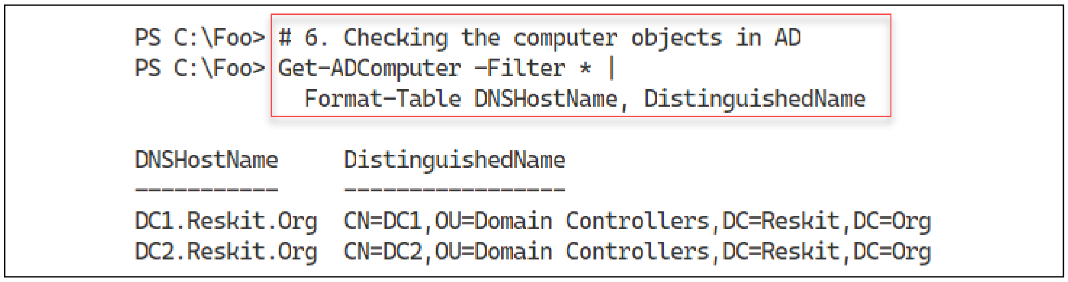


```
PS C:\Foo> # 4. Adding the AD DS features on DC2
PS C:\Foo> Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools

Success Restart Needed Exit Code      Feature Result
------- -------------- ---------      --------------
True    No             Success        {Active Directory Domain Services, Group Pol…
```

Figure 6.20: Adding the AD DS features on DC2

With all the prerequisites in place, in *step 5*, you promote `DC2` to be a DC and ensure that the DC does not reboot after this step completes. There is no output from this step.

In *step 6*, before rebooting `DC2` (necessary to complete the promotion process), you check to see the computer objects now in AD. This step helps you to ensure that `DC2` is now in the correct place in AD. The output of this step is like this:

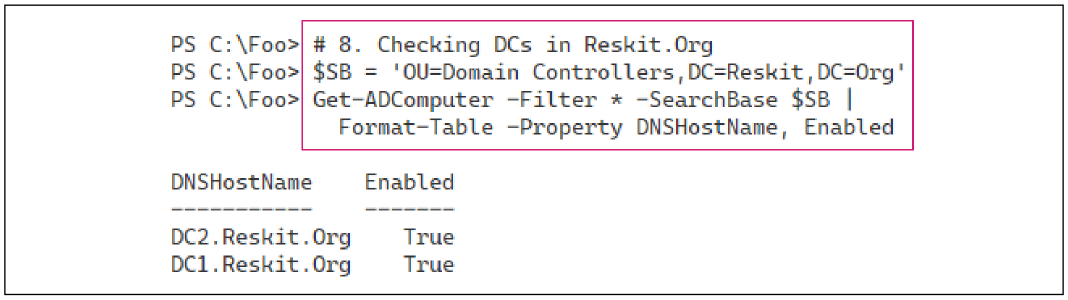


```
PS C:\Foo> # 6. Checking the computer objects in AD
PS C:\Foo> Get-ADComputer -Filter * |
             Format-Table DNSHostName, DistinguishedName

DNSHostName     DistinguishedName
-----------     -----------------
DC1.Reskit.Org  CN=DC1,OU=Domain Controllers,DC=Reskit,DC=Org
DC2.Reskit.Org  CN=DC2,OU=Domain Controllers,DC=Reskit,DC=Org
```

Figure 6.21: Checking the computer objects in the AD

In *step 7*, you finalize the promoting process and reboot `DC2`. Once rebooted, you need to log on as the domain administrator (`Reskit\Administrator`). This step produces no output as such.

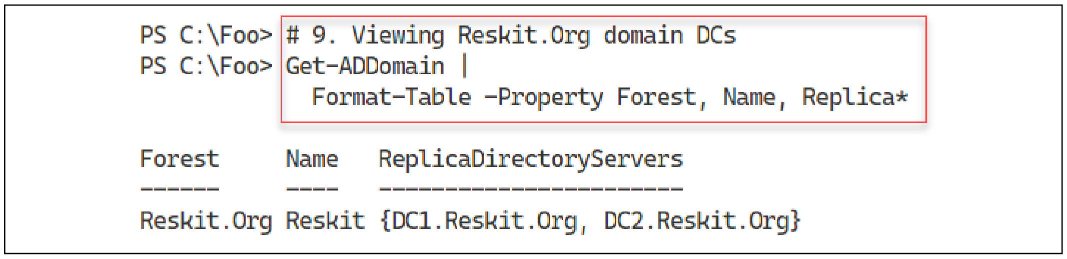
With *step 8*, you examine the hosts in the Domain Controllers OU, which looks like this:



```
PS C:\Foo> # 8. Checking DCs in Reskit.Org
PS C:\Foo> $SB = 'OU=Domain Controllers,DC=Reskit,DC=Org'
PS C:\Foo> Get-ADComputer -Filter * -SearchBase $SB |
             Format-Table -Property DNSHostName, Enabled

DNSHostName    Enabled
-----------    -------
DC2.Reskit.Org    True
DC1.Reskit.Org    True
```

Figure 6.22: Checking DCs in Reskit.Org

In the final step in this recipe, *step 9*, you use the `Get-ADDomain` cmdlet to check that `DC1` and `DC2` are DCs for this domain. The output of this step looks like this:



```
PS C:\Foo> # 9. Viewing Reskit.Org domain DCs
PS C:\Foo> Get-ADDomain |
             Format-Table -Property Forest, Name, Replica*

Forest     Name   ReplicaDirectoryServers
------     ----   -----------------------
Reskit.Org Reskit {DC1.Reskit.Org, DC2.Reskit.Org}
```

Figure 6.23: Viewing Reskit.Org domain DCs

There's more...

In *step 1*, you import the `ServerManager` module manually. As you have seen in earlier recipes, this module is not supported natively with PowerShell 7. This step loads the module using the Windows PowerShell Compatibility solution described earlier in the book.

In *step 3*, you check to ensure `DC2` can connect to `DC1` over ports `445` and `389`. Windows uses TCP port `445` for SMB file sharing. The group policy agent on each domain-joined host uses this port to retrieve group policy details from the SYSVOL share on `DC1`. LDAP uses port `389` to perform actions against a DC, such as adding a new user or checking group membership. Both ports need to be open and contactable if the promotion of `DC2` is to be successful.

In *step 6*, you retrieve all computer accounts currently in your AD. By default, the AD DC promotion process ensures that this host's computer account is now in the Domain Controllers OU. The location of a DC in an OU is essential as it enables your new DC to apply the group policy settings on this OU. If `DC2` were a workgroup computer and not joined to the domain, the promotion process would create this account in the Domain Controllers OU. If `DC2` were a domain member, then the promotion process would move the computer account into the Domain Controllers OU.

In this recipe, before promoting `DC2` to be a DC, you check to ensure that the prerequisites are in place before the promotion. Then you ensure that you have configured your forest, domain, and DCs correctly. Mistakes are easy to make, but also (usually) easy to correct. In practice, this is redundant checking since most AD promotions "just work." The additional checks ensure you discover and resolve any issues that might affect the promotion or cause the AD to not work correctly after a not-fully-successful promotion.

Installing a child domain

As you saw in *Installing a replica domain controller*, adding a DC to an existing domain is straightforward. So long as prerequisites like DNS are in place, the promotion process is simple.

An AD forest can contain more than one domain. This architecture has some value in terms of delegated control and some reduction in replication traffic. And like creating a replica DC, creating a child domain is simple, as you can see in this recipe.

Best practice calls for a contiguous namespace of domains, where the additional domain is a child of another existing domain. In this recipe, you create a child domain to the `Reskit.Org` domain you created earlier. You promote the server `UKDC1` to be the first DC in a new child domain, `UK.Reskit.Org`. The steps in this recipe are very similar to those in *Installing a replica domain controller*.

Getting ready

You run this recipe on `UKDC1`, a domain-joined server in the `Reskit.Org` domain on which you have installed PowerShell 7 and VS Code. You create this server after you have installed `DC1` as a DC.

How to do it...

1. Importing the `ServerManager` module

   ```
   Import-Module -Name ServerManager -WarningAction SilentlyContinue
   ```

2. Checking `DC1` can be resolved

   ```
   Resolve-DnsName -Name DC1.Reskit.Org -Type A
   ```

3. Checking network connection to `DC1`

   ```
   Test-NetConnection -ComputerName DC1.Reskit.Org -Port 445
   Test-NetConnection -ComputerName DC1.Reskit.Org -Port 389
   ```

4. Adding the AD DS features on `UKDC1`

   ```
   $Features = 'AD-Domain-Services'
   Install-WindowsFeature -Name $Features -IncludeManagementTools
   ```

5. Creating a credential and installation hash table

   ```
   Import-Module -Name ADDSDeployment -WarningAction SilentlyContinue
   $URK    = 'Administrator@Reskit.Org'
   ```

```
$PW     = 'Pa$$w0rd'
$PSS    = ConvertTo-SecureString -String $PW -AsPlainText -Force
$CredRK = [PSCredential]::New($URK,$PSS)
$INSTALLHT    = @{
  NewDomainName                 = 'UK'
  ParentDomainName              = 'Reskit.Org'
  DomainType                    = 'ChildDomain'
  SafeModeAdministratorPassword = $PSS
  ReplicationSourceDC           = 'DC1.Reskit.Org'
  Credential                    = $CredRK
  SiteName                      = 'Default-First-Site-Name'
  InstallDNS                    = $false
  Force                         = $true
}
```

6. Installing child domain

```
Install-ADDSDomain @INSTALLHT
```

7. Looking at the AD forest

```
Get-ADForest -Server UKDC1.UK.Reskit.Org
```

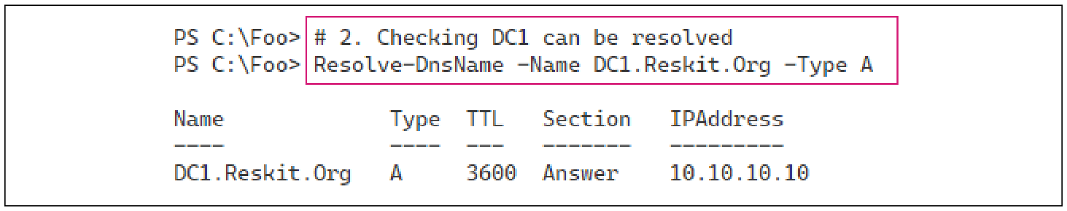
8. Looking at the UK domain

```
Get-ADDomain -Server UKDC1.UK.Reskit.Org
```

How it works...

In *step 1*, you load the `ServerManager` module. This module is not natively supported by PowerShell 7. This step, which produces no output, loads the module using the Windows PowerShell Compatibility solution.

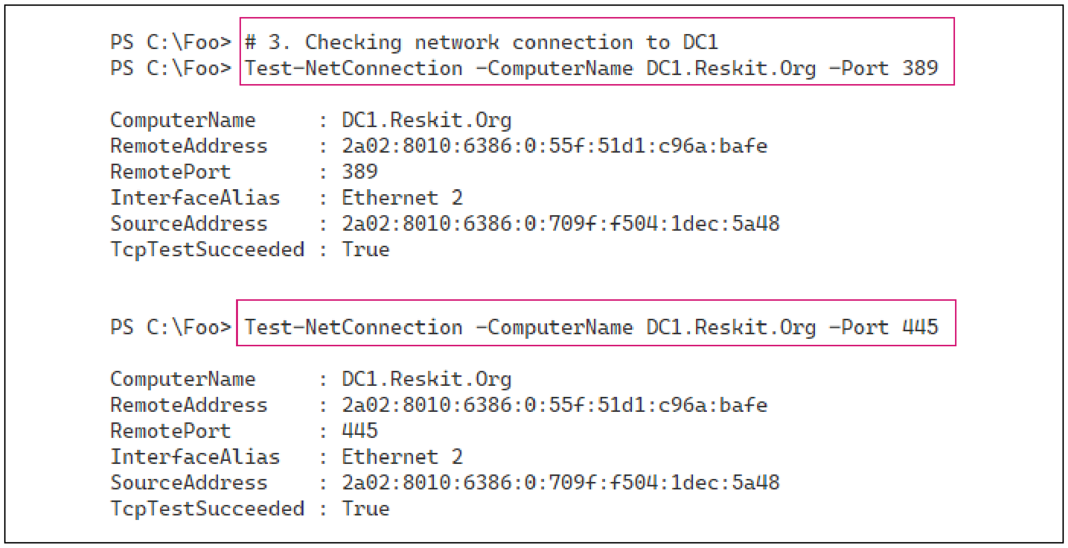
When you create a new domain and new DC (using `UKDC1`, for example), the server needs to contact the holder of the domain naming FSMO role, the DC responsible for the forest's changes. In *step 2,* you check to ensure you can reach the host, `DC1`, which looks like this:



```
PS C:\Foo> # 2. Checking DC1 can be resolved
PS C:\Foo> Resolve-DnsName -Name DC1.Reskit.Org -Type A

Name             Type   TTL   Section    IPAddress
----             ----   ---   -------    ---------
DC1.Reskit.Org   A      3600  Answer     10.10.10.10
```

Figure 6.24: Checking DC1 can be resolved

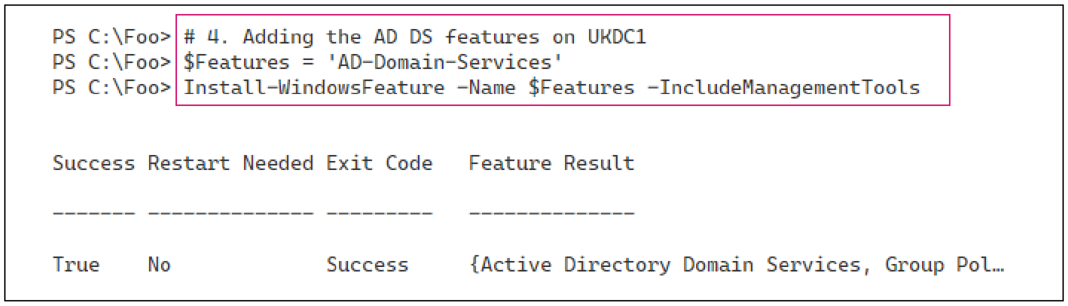
In *step 3*, you check that you can connect to `DC1` on ports `445` and `389`, both required for proper domain functioning. The output of this step looks like this:



```
PS C:\Foo> # 3. Checking network connection to DC1
PS C:\Foo> Test-NetConnection -ComputerName DC1.Reskit.Org -Port 389

ComputerName     : DC1.Reskit.Org
RemoteAddress    : 2a02:8010:6386:0:55f:51d1:c96a:bafe
RemotePort       : 389
InterfaceAlias   : Ethernet 2
SourceAddress    : 2a02:8010:6386:0:709f:f504:1dec:5a48
TcpTestSucceeded : True


PS C:\Foo> Test-NetConnection -ComputerName DC1.Reskit.Org -Port 445

ComputerName     : DC1.Reskit.Org
RemoteAddress    : 2a02:8010:6386:0:55f:51d1:c96a:bafe
RemotePort       : 445
InterfaceAlias   : Ethernet 2
SourceAddress    : 2a02:8010:6386:0:709f:f504:1dec:5a48
TcpTestSucceeded : True
```

Figure 6.25: Checking network connection to DC1

In *step 4*, you add the AD DS feature to `UKDC1`. This feature is required to install the necessary tools you use to create the child domain. The output of this step looks like this:

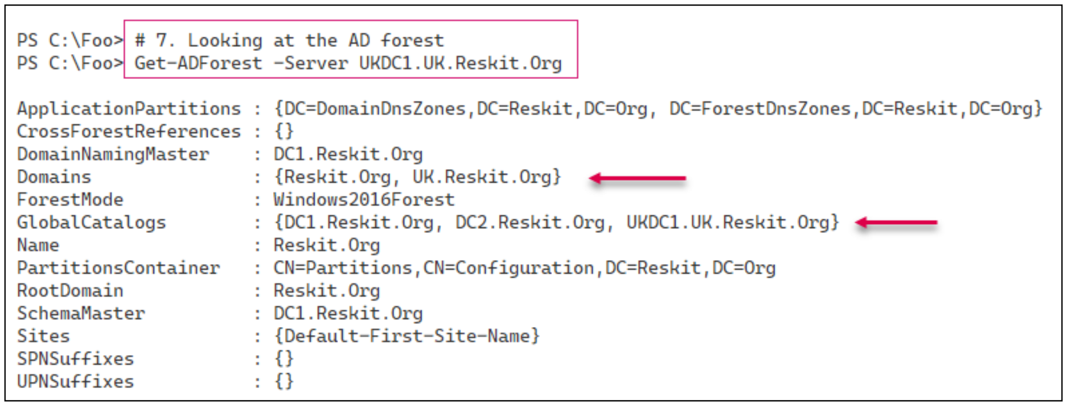


```
PS C:\Foo> # 4. Adding the AD DS features on UKDC1
PS C:\Foo> $Features = 'AD-Domain-Services'
PS C:\Foo> Install-WindowsFeature -Name $Features -IncludeManagementTools


Success Restart Needed Exit Code   Feature Result

------- -------------- ---------   --------------

True    No             Success     {Active Directory Domain Services, Group Pol…
```

Figure 6.26: Adding the AD DS features on UKDC1

In *step 5*, you build a hash table of parameters that you use in a later step to perform the promotion. In *step 6*, you use this hash table when calling `Install-ADDSDomain` to create a child domain. These two steps create no output. After the promotion has completed, your host reboots; log on as `Reskit\Administrator`.

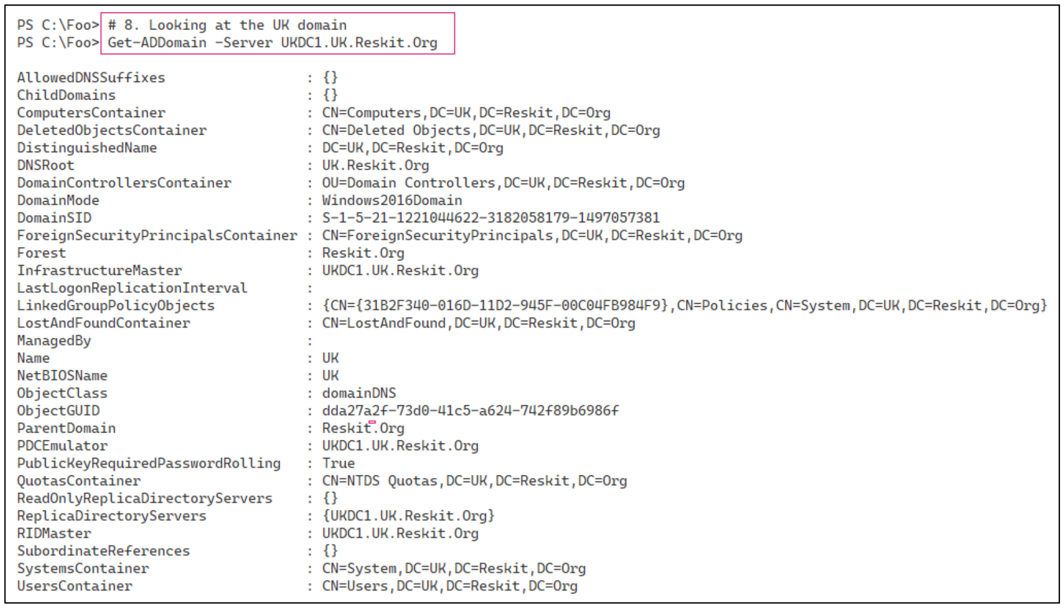
Once you have logged in, you can check the AD forest details from your newly promoted child domain DC. In *step 7*, you use `Get-ADForest`, with output like this:



```
PS C:\Foo> # 7. Looking at the AD forest
PS C:\Foo> Get-ADForest -Server UKDC1.UK.Reskit.Org

ApplicationPartitions : {DC=DomainDnsZones,DC=Reskit,DC=Org, DC=ForestDnsZones,DC=Reskit,DC=Org}
CrossForestReferences : {}
DomainNamingMaster    : DC1.Reskit.Org
Domains               : {Reskit.Org, UK.Reskit.Org}
ForestMode            : Windows2016Forest
GlobalCatalogs        : {DC1.Reskit.Org, DC2.Reskit.Org, UKDC1.UK.Reskit.Org}
Name                  : Reskit.Org
PartitionsContainer   : CN=Partitions,CN=Configuration,DC=Reskit,DC=Org
RootDomain            : Reskit.Org
SchemaMaster          : DC1.Reskit.Org
Sites                 : {Default-First-Site-Name}
SPNSuffixes           : {}
UPNSuffixes           : {}
```

Figure 6.27: Looking at the AD forest details

In *step 8*, you examine the domain details of the newly created child domain. By using `Get-ADDomain` and targeting the new DC, you see output that looks like this:



```
PS C:\Foo> # 8. Looking at the UK domain
PS C:\Foo> Get-ADDomain -Server UKDC1.UK.Reskit.Org

AllowedDNSSuffixes                 : {}
ChildDomains                       : {}
ComputersContainer                 : CN=Computers,DC=UK,DC=Reskit,DC=Org
DeletedObjectsContainer            : CN=Deleted Objects,DC=UK,DC=Reskit,DC=Org
DistinguishedName                  : DC=UK,DC=Reskit,DC=Org
DNSRoot                            : UK.Reskit.Org
DomainControllersContainer         : OU=Domain Controllers,DC=UK,DC=Reskit,DC=Org
DomainMode                         : Windows2016Domain
DomainSID                          : S-1-5-21-1221044622-3182058179-1497057381
ForeignSecurityPrincipalsContainer : CN=ForeignSecurityPrincipals,DC=UK,DC=Reskit,DC=Org
Forest                             : Reskit.Org
InfrastructureMaster               : UKDC1.UK.Reskit.Org
LastLogonReplicationInterval       :
LinkedGroupPolicyObjects           : {CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=UK,DC=Reskit,DC=Org}
LostAndFoundContainer              : CN=LostAndFound,DC=UK,DC=Reskit,DC=Org
ManagedBy                          :
Name                               : UK
NetBIOSName                        : UK
ObjectClass                        : domainDNS
ObjectGUID                         : dda27a2f-73d0-41c5-a624-742f89b6986f
ParentDomain                       : Reskit.Org
PDCEmulator                        : UKDC1.UK.Reskit.Org
PublicKeyRequiredPasswordRolling   : True
QuotasContainer                    : CN=NTDS Quotas,DC=UK,DC=Reskit,DC=Org
ReadOnlyReplicaDirectoryServers    : {}
ReplicaDirectoryServers            : {UKDC1.UK.Reskit.Org}
RIDMaster                          : UKDC1.UK.Reskit.Org
SubordinateReferences              : {}
SystemsContainer                   : CN=System,DC=UK,DC=Reskit,DC=Org
UsersContainer                     : CN=Users,DC=UK,DC=Reskit,DC=Org
```

Figure 6.28: Looking at the UK domain details

There's more...

In *step 2*, you check that `DC1` is the domain naming FSMO role holder. Assuming you have installed the `Reskit.Org` domain and forest as shown in this chapter, that role holder is `DC1`. You could call `Get-ADForest` and obtain the hostname from the `DomainNamingMaster` property.

In *steps 5* and *6*, you promote `UKDC1` to be the first DC in a new child domain, `UK.Reskit.Org`. Unlike in the two previous DC promotions (that is, promoting `DC1` and `DC2`), in this step, you allow Windows to reboot immediately after the promotion has completed. This step can take quite a while – potentially 10 minutes or more – so be patient.

In *step 7*, you use `Get-ADForest` to examine the details of the forest as stored on `UKDC1`. As you can see in *Figure 6.27*, these details now show your new domain (`UK.Reskit.Org`) in the `Domains` property. Also, by default, you can see that `UKDC1.UK.Reskit.Org` is also a global catalog server.

Creating and managing AD users and groups

After creating your forest/domain and your DCs, you can begin to manage the core objects in AD, namely, users, groups, computers, and organizational units. User and computer accounts identify a specific user or computer. Windows uses these objects to enable the computer and the user to log on securely using passwords held in the AD.

AD groups enable you to collect users and computers into a single (group) account that simplifies setting access controls on resources such as files or file shares. As you saw in *Testing an AD installation*, when you create a new forest, the AD promotion process creates many potentially useful groups.

Organizational units enable you to partition users, computers, and groups into separate container OUs. OUs provide you with essential roles in your AD. The first is role delegation. You can delegate the management of any OU (and child OUs) to be carried out by different groups. For example, you could create a top-level OU called `UK` in the `Reskit.Org` domain. You could then delegate permissions to the objects in this OU to a group, such as `UKAdmins`, enabling any group member to manage AD objects in, and below, the UK OU.

The second role played by OUs is to act as a target for group policy objects. You could create a GPO for the IT team and apply it to the IT OU. You could create a separate OU and create GPOs that apply to only the computer and user objects in that OU. Thus, each user and computer in a given OU is configured based on the GPO.

In this recipe, you examine AD user and group objects. In a later recipe, *Reporting on AD computers*, you explore managing AD computers. And in *Creating group policy objects*, you assign a group policy to an OU you create in this recipe.

Getting ready

You run this recipe on DC1, a DC in the Reskit.Org domain. You have a workgroup server on which you have installed PowerShell 7 and VS Code on this host.

How to do it...

1. Creating a hash table for general user attributes

```
$PW  = 'Pa$$w0rd'
$PSS = ConvertTo-SecureString -String $PW -AsPlainText -Force
$NewUserHT = @{}
$NewUserHT.AccountPassword       = $PSS
$NewUserHT.Enabled               = $true
$NewUserHT.PasswordNeverExpires  = $true
$NewUserHT.ChangePasswordAtLogon = $false
```

2. Creating two new users

```
# First user
$NewUserHT.SamAccountName    = 'ThomasL'
$NewUserHT.UserPrincipalName = 'thomasL@reskit.org'
$NewUserHT.Name              = 'ThomasL'
$NewUserHT.DisplayName       = 'Thomas Lee (IT)'
New-ADUser @NewUserHT
# Second user
$NewUserHT.SamAccountName    = 'RLT'
$NewUserHT.UserPrincipalName = 'rlt@reskit.org'
$NewUserHT.Name              = 'Rebecca Tanner'
$NewUserHT.DisplayName       = 'Rebecca Tanner (IT)'
New-ADUser @NewUserHT
```

3. Creating an OU for IT

```
$OUHT = @{
    Name        = 'IT'
    DisplayName = 'Reskit IT Team'
    Path        = 'DC=Reskit,DC=Org'
}
New-ADOrganizationalUnit @OUHT
```

4. Moving users into the OU

```
$MHT1 = @{
    Identity   = 'CN=ThomasL,CN=Users,DC=Reskit,DC=ORG'
    TargetPath = 'OU=IT,DC=Reskit,DC=Org'
}
```

```
Move-ADObject @MHT1
$MHT2 = @{
    Identity = 'CN=Rebecca Tanner, CN=Users, DC=Reskit, DC=ORG'
    TargetPath = 'OU=IT,DC=Reskit,DC=Org'
}
Move-ADObject @MHT2
```

5. Creating a third user directly in the IT OU

```
$NewUserHT.SamAccountName    = 'JerryG'
$NewUserHT.UserPrincipalName = 'jerryg@reskit.org'
$NewUserHT.Description       = 'Virtualization Team'
$NewUserHT.Name              = 'Jerry Garcia'
$NewUserHT.DisplayName       = 'Jerry Garcia (IT)'
$NewUserHT.Path              = 'OU=IT,DC=Reskit,DC=Org'
New-ADUser @NewUserHT
```

6. Adding two users who get removed later

```
# First user to be removed
$NewUserHT.SamAccountName    = 'TBR1'
$NewUserHT.UserPrincipalName = 'tbr@reskit.org'
$NewUserHT.Name              = 'TBR1'
$NewUserHT.DisplayName       = 'User to be removed'
$NewUserHT.Path              = 'OU=IT,DC=Reskit,DC=Org'
New-ADUser @NewUserHT
# Second user to be removed
$NewUserHT.SamAccountName     = 'TBR2'
$NewUserHT.UserPrincipalName  = 'tbr2@reskit.org'
$NewUserHT.Name               = 'TBR2'
New-ADUser @NewUserHT
```

7. Viewing existing AD users

```
Get-ADUser -Filter *  -Property *|
  Format-Table -Property Name, Displayname, SamAccountName
```

8. Removing a user via a `Get | Remove` pattern

```
Get-ADUser -Identity 'CN=TBR1,OU=IT,DC=Reskit,DC=Org' |
    Remove-ADUser -Confirm:$false
```

9. Removing a user directly

```
$RUHT = @{
  Identity = 'CN=TBR2,OU=IT,DC=Reskit,DC=Org'
  Confirm  = $false}
Remove-ADUser @RUHT
```

10. Updating a user object

```
$TLHT =@{
  Identity     = 'ThomasL'
  OfficePhone  = '4416835420'
  Office       = 'Cookham HQ'
  EmailAddress = 'ThomasL@Reskit.Org'
  GivenName    = 'Thomas'
  Surname      = 'Lee'
  HomePage     = 'Https://tfl09.blogspot.com'
}
Set-ADUser @TLHT
```

11. Viewing updated user

```
Get-ADUser -Identity ThomasL -Properties * |
  Format-Table -Property DisplayName,Name,Office,
             OfficePhone,EmailAddress
```

12. Creating a new domain local group

```
$NGHT = @{
 Name        = 'IT Team'
 Path        = 'OU=IT,DC=Reskit,DC=org'
 Description = 'All members of the IT Team'
 GroupScope  = 'DomainLocal'
}
New-ADGroup @NGHT
```

13. Adding all the users in the IT OU into the IT Team group

```
$SB = 'OU=IT,DC=Reskit,DC=Org'
$ItUsers = Get-ADUser -Filter * -SearchBase $SB
Add-ADGroupMember -Identity 'IT Team' -Members $ItUsers
```

14. Displaying members of the IT Team group

```
Get-ADGroupMember -Identity 'IT Team' |
  Format-Table SamAccountName, DistinguishedName
```

How it works...

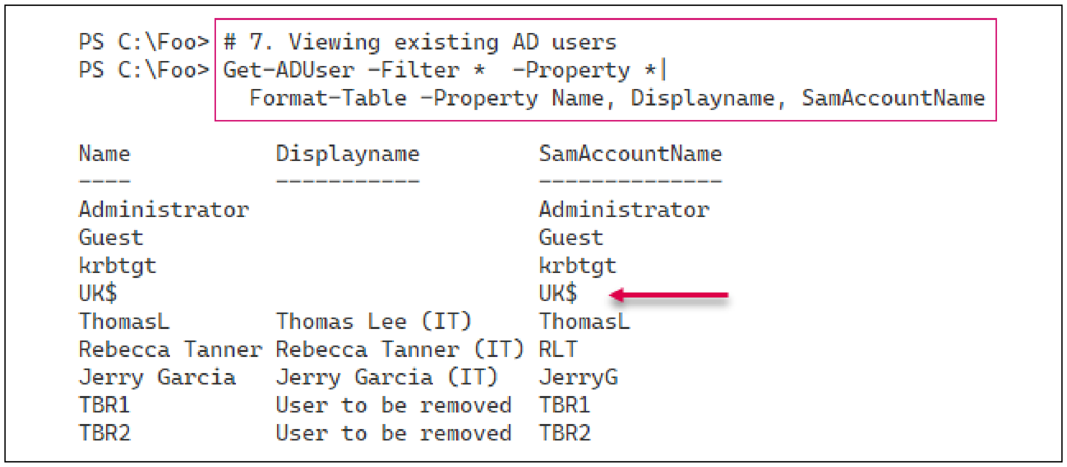
You use the `New-ADUser` cmdlet to create a new AD user. Due to the amount of information you wish to hold for any user you create, the number of parameters you might need to pass to `New-ADUser` can make for very long code lines. To get around that, you create a hash table of parameters and parameter values and then use it to create the user. In *step 1*, you create such a hash table, which creates no output.

In *step 2*, you add to the hash table you created in the previous step and create two new AD users. This step creates no output.

With *step 3*, you create a new OU, `IT`, which creates no output. You use this OU to collect user and computer objects for the IT department of Reskit.Org. Next, in *step 4*, which also creates no output, you move the two users you created previously into the IT OU.

Rather than creating a user (which, by default, places the new user object in the Users container in your AD) and later moving it to an OU, you can create a new user object directly in an OU. In *step 5*, you create a third new user directly in the IT OU, which creates no output.

In *step 6*, which generates no output, you create two additional users, which you use later in the recipe. With *step 7*, you use `Get-ADUser` to retrieve all the user objects in the `Reskit.Org` domain, which looks like this:

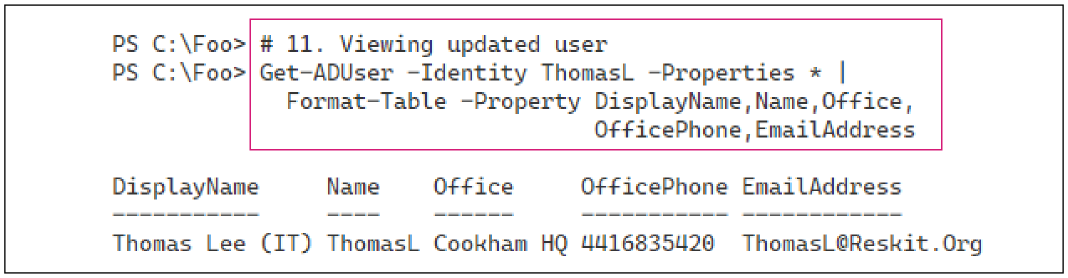


```
PS C:\Foo> # 7. Viewing existing AD users
PS C:\Foo> Get-ADUser -Filter *  -Property *|
             Format-Table -Property Name, Displayname, SamAccountName

Name            Displayname          SamAccountName
----            -----------          --------------
Administrator                        Administrator
Guest                                Guest
krbtgt                               krbtgt
UK$                                  UK$
ThomasL         Thomas Lee (IT)      ThomasL
Rebecca Tanner  Rebecca Tanner (IT)  RLT
Jerry Garcia    Jerry Garcia (IT)    JerryG
TBR1            User to be removed   TBR1
TBR2            User to be removed   TBR2
```

Figure 6.29: Viewing existing AD users

You can remove any AD user by invoking `Remove-ADUser`. There are two broad patterns for removing a user. The first pattern involves using `Get-ADUser` to get the user you want to remove and then piping that to `Remove-ADUser`. This pattern is appropriate for the command line, where you should always verify you have the right user before deleting that user. The second, possibly more appropriate in scripts, is to remove the user in one operation (and deal with any risks of removing the wrong user). In *step 8*, you use the `Get | Remove` pattern to get the user, then remove it. In *step 9*, you remove the user directly by specifying the user's identity when calling `Remove-ADUser`. Neither *step 8* nor *step 9* generate output.

In *step 10*, you update a user object with new/updated properties. This step generates no output. To see the changes, in *step 11*, you can use `Get-ADUser` and view the properties of the updated user, which looks like this:



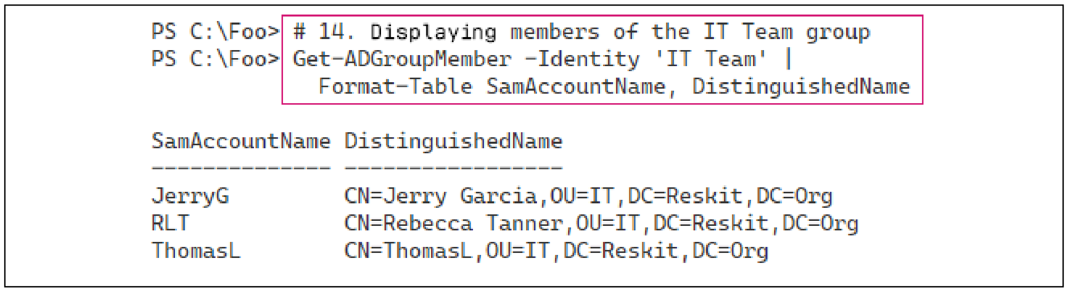
```
PS C:\Foo> # 11. Viewing updated user
PS C:\Foo> Get-ADUser -Identity ThomasL -Properties * |
             Format-Table -Property DisplayName,Name,Office,
                                    OfficePhone,EmailAddress

DisplayName     Name    Office     OfficePhone EmailAddress
-----------     ----    ------     ----------- ------------
Thomas Lee (IT) ThomasL Cookham HQ 4416835420  ThomasL@Reskit.Org
```

Figure 6.30: Viewing the updated user

AD enables two types of group account, both of which can contain users and groups: security and distribution. You use distribution groups typically for email systems. Exchange, for example, uses them to implement distribution lists. You use security groups to assign permissions and rights. For example, if you allow a group to have access to a file or folder, then by default, all members of that group have that access permission. Using groups is a best practice when setting permissions and rights.

Security groups in AD have three scopes, which offer different features in terms of members and how you use them to assign permissions and rights. For more details on the different scopes and AD security groups in general, see `https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/active-directory-security-groups`.

In *step 12*, you create a new domain local group, `IT Team`. In *step 13*, you add all the users in the IT OU to this IT Team group. Neither step produces output. In *step 14*, you get and display the members of the IT Team group, which looks like this:



```
PS C:\Foo> # 14. Displaying members of the IT Team group
PS C:\Foo> Get-ADGroupMember -Identity 'IT Team' |
             Format-Table SamAccountName, DistinguishedName

SamAccountName DistinguishedName
-------------- -----------------
JerryG         CN=Jerry Garcia,OU=IT,DC=Reskit,DC=Org
RLT            CN=Rebecca Tanner,OU=IT,DC=Reskit,DC=Org
ThomasL        CN=ThomasL,OU=IT,DC=Reskit,DC=Org
```

Figure 6.31: Displaying members of the IT Team group

There's more...

In *step 7*, you see the users in the Reskit.Org domain. Note the user `UK$`. This user relates to the child domain (`UK.Reskit.Org`). It is not, as such, an actual user. The `$` character at the end of the username indicates it's a hidden user account but fundamental to supporting the child domain. Don't be tempted to tidy up and remove this user account – that would break the child domain structure.

As you see in this recipe, the cmdlets you use to add or modify a user, group, or OU create no output, which cuts down on the output you have to wade through. Some cmdlets and cmdlet sets would output details of the objects created, updated, or possibly deleted. Consider the lack of output for AD cmdlets as a feature.

Managing AD computers

AD computer objects represent domain-joined computers that can use the domain to authenticate user login. Before you can log in as a domain user, such as `Reskit\JerryG`, your computer must be a domain member. When a domain-joined computer starts up, it contacts a DC to authenticate itself. In effect, the computer logs into the domain and creates a secure channel to the DC. Once Windows establishes this secure channel, Windows can log on a user. Under the covers, Windows uses the secure channel to negotiate the user logon.

In this recipe, you work with AD computers and add `SRV1` to the `Reskit.Org` domain.

Getting ready

You run this recipe on `DC1`, a DC in the Reskit.Org domain. This recipe also uses `SRV1`. This host starts as a non-domain-joined Windows Server 2022 host (which you used in earlier chapters in this book). You also use `UKDC1` (the DC in the `UK.Reskit.Org` domain). You should have PowerShell 7 and VS Code installed on each of these hosts.

How to do it...

1. Getting computers in the Reskit domain

   ```
   Get-ADComputer -Filter * |
     Format-Table -Property Name, DistinguishedName
   ```

2. Getting computers in the UK domain

   ```
   Get-ADComputer -Filter * -Server UKDC1.UK.Reskit.Org |
     Format-Table -Property Name, DistinguishedName
   ```

3. Creating a new computer in the `Reskit.Org` domain

```
$NCHT = @{
    Name                   = 'Wolf'
    DNSHostName            = 'Wolf.Reskit.Org'
    Description            = 'One for Jerry'
    Path                   = 'OU=IT,DC=Reskit,DC=Org'
}
New-ADComputer @NCHT
```

4. Creating a credential object for `SRV1`

```
$ASRV1    = 'SRV1\Administrator'
$PSRV1    = 'Pa$$w0rd'
$PSSRV1   = ConvertTo-SecureString -String $PSRV1 -AsPlainText -Force
$CredSRV1 = [pscredential]::New($ASRV1, $PSSRV1)
```

5. Creating a script block to join `SRV1`

```
$SB = {
  $ARK    = 'Reskit\Administrator'
  $PRK    = 'Pa$$w0rd'
  $PSRK   = ConvertTo-SecureString -String $PRK -AsPlainText -Force
  $CredRK = [pscredential]::New($ARK, $PSRK)
  $DJHT = @{
    DomainName  = 'Reskit.Org'
    OUPath      = 'OU=IT,DC=Reskit,DC=Org'
    Credential  = $CredRK
    Restart     = $false
 }
    Add-Computer @DJHT
}
```

6. Joining the computer to the domain

```
Set-Item -Path WSMan:\localhost\Client\TrustedHosts -Value '*'
Invoke-Command -ComputerName SRV1 -Credential $CredSRV1 -ScriptBlock $SB
```

7. Restarting `SRV1`

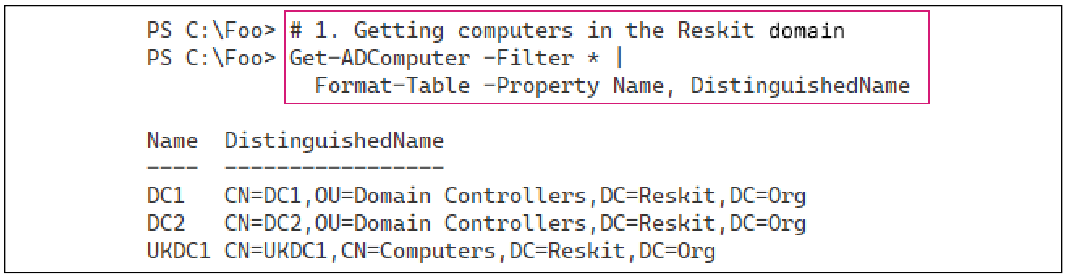
```
Restart-Computer -ComputerName SRV1 -Credential $CredSRV1 -Force
```

8. Viewing the resulting computer accounts for `Reskit.Org`

```
Get-ADComputer -Filter * -Properties DNSHostName,LastLogonDate |
  Format-Table -Property Name, DNSHostName, Enabled
```

How it works...

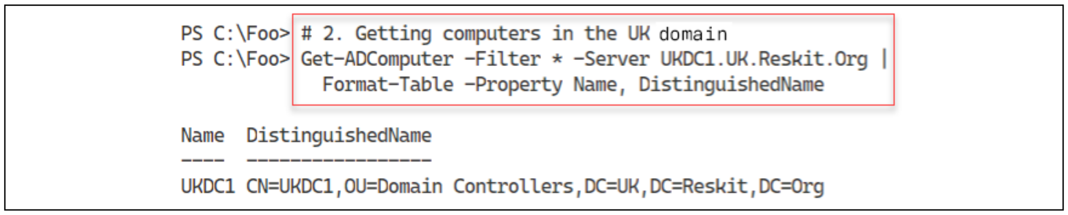
In *step 1*, you use the `Get-ADComputer` cmdlet to obtain the computer objects defined thus far in the AD, which looks like this:



```
PS C:\Foo> # 1. Getting computers in the Reskit domain
PS C:\Foo> Get-ADComputer -Filter * |
             Format-Table -Property Name, DistinguishedName

Name  DistinguishedName
----  -----------------
DC1   CN=DC1,OU=Domain Controllers,DC=Reskit,DC=Org
DC2   CN=DC2,OU=Domain Controllers,DC=Reskit,DC=Org
UKDC1 CN=UKDC1,CN=Computers,DC=Reskit,DC=Org
```

Figure 6.32: Getting computers in the Reskit domain

In *step 2*, you obtain the computers in the `UK.Reskit.Org` domain from that domain's DC, `UKDC1`, which looks like this:



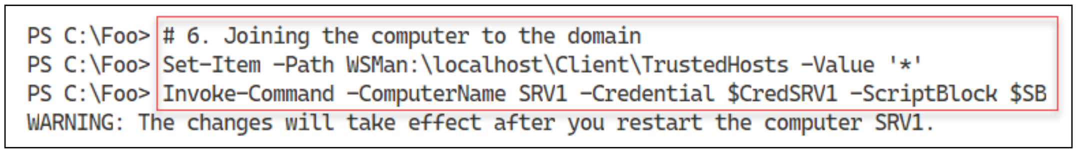
```
PS C:\Foo> # 2. Getting computers in the UK domain
PS C:\Foo> Get-ADComputer -Filter * -Server UKDC1.UK.Reskit.Org |
             Format-Table -Property Name, DistinguishedName

Name  DistinguishedName
----  -----------------
UKDC1 CN=UKDC1,OU=Domain Controllers,DC=UK,DC=Reskit,DC=Org
```

Figure 6.33: Getting computers in the UK domain

In *step 3*, you create a new computer in the Reskit domain, `Wolf`, or `Wolf.Reskit.Org`. This step creates no output.

To prepare to add `SRV1` to the domain in a single operation, in *step 4*, you create a credential object for `SRV1`'s administrator user. For this credential, you use the generic password used throughout this book. Feel free to use your own password scheme. You need this credential object because `SRV1` is currently a workgroup computer. In *step 5*, you create a script block that, when you run it, adds a computer to the `Reskit.Org` domain. Neither step produces any output.

In *step 6*, you invoke the script block on `SRV1`, which adds `SRV1` to the `Reskit.Org` domain. The output of this step looks like this:

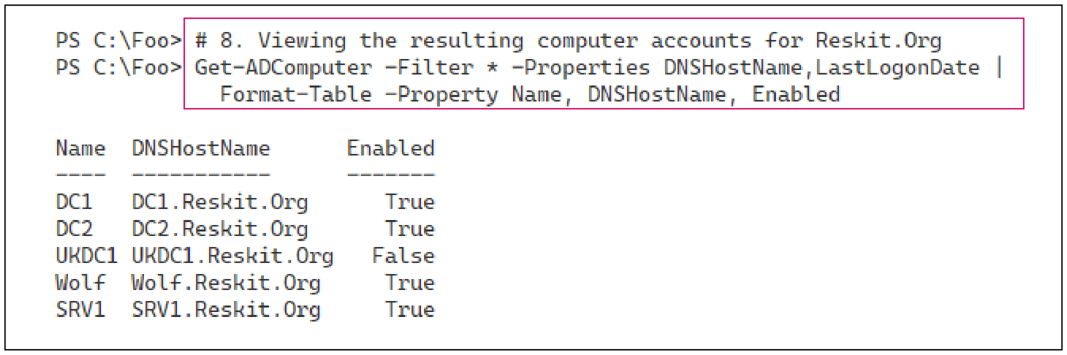


```
PS C:\Foo> # 6. Joining the computer to the domain
PS C:\Foo> Set-Item -Path WSMan:\localhost\Client\TrustedHosts -Value '*'
PS C:\Foo> Invoke-Command -ComputerName SRV1 -Credential $CredSRV1 -ScriptBlock $SB
WARNING: The changes will take effect after you restart the computer SRV1.
```

Figure 6.34: Joining the computer to the domain

To complete the process of joining `SRV1` to the Reskit.Org domain, in *step 7*, you reboot `SRV1`, which creates no output.

After `SRV1` has completed its reboot, in *step 8,* you view all the accounts now in the Reskit.Org domain, which looks like this:



```
PS C:\Foo> # 8. Viewing the resulting computer accounts for Reskit.Org
PS C:\Foo> Get-ADComputer -Filter * -Properties DNSHostName,LastLogonDate |
             Format-Table -Property Name, DNSHostName, Enabled

Name  DNSHostName      Enabled
----  -----------      -------
DC1   DC1.Reskit.Org      True
DC2   DC2.Reskit.Org      True
UKDC1 UKDC1.Reskit.Org   False
Wolf  Wolf.Reskit.Org     True
SRV1  SRV1.Reskit.Org     True
```

Figure 6.35: Viewing resulting computer accounts for Reskit.Org

There's more...

There are two broad ways of adding a computer to a domain. The first is to log on to the computer to be added and join the domain. To achieve this, you must have credentials for a user with permissions needed to add a computer to a domain (that is, the domain administrator). You also need the credentials that enable you to log on to the system itself. Alternatively, you can create a computer object in advance, which is known as **pre-staging**. You need administrator credentials for this operation, but once pre-staged any user can join the computer to the domain.

In *step 3*, you pre-stage the `Wolf` computer. A user able to log on to `Wolf` could then use `Add-Computer` (or the GUI) to add the host to the domain. In *step 4*, you add a computer to a domain using domain administrator credentials.

Adding/removing users using CSV files

Spiceworks (`https://www.spiceworks.com/`) is an excellent site for IT professionals to learn more and get their problems solved. Spiceworks has a busy PowerShell support forum, which you can access at `https://community.spiceworks.com/programming/powershell`.

A frequently asked (and answered) question is: How do I add multiple users using an input file? This recipe does just that. You start with a CSV file containing details of the users you are going to add. Then you run this recipe to add the users.

This recipe uses a CSV file of users to add to AD, with a limited set of properties and values. In production, you would most likely extend the information contained in the CSV, based on your business needs and the information you want to store in AD.

Getting ready

You run this recipe on `SRV1`, a domain-joined server on which you have installed PowerShell 7 and VS Code. Log in as `Reskit\Administrator`. You should also have `DC1` and `DC2` up and running.

This recipe creates and uses a CSV file. As an alternative to using *step 1* in the recipe, you can also download the CSV from GitHub at `https://github.com/doctordns/PACKT-PS7/blob/master/scripts/Ch%206%20-%20AD/goodies/users.csv`. If you download it from GitHub, make sure you store it in `C:\Foo\Users.CSV`.

How to do it...

1. Creating a CSV file

```
$CSVDATA = @'
Firstname, Initials,Lastname,UserPrincipalName,Alias,Description,
Password
J, K,Smith, JKS, James, Data Team, Christmas42
Clair, B, Smith, CBS, Claire, Receptionist, Christmas42
Billy, Bob, JoeBob, BBJB, BillyBob, A Bob, Christmas42
Malcolm, Dudley, Duewrong, Malcolm, Malcolm, Mr Danger, Christmas42
'@
$CSVDATA | Out-File -FilePath C:\Foo\Users.Csv
```

2. Importing and displaying the CSV

```
$Users = Import-CSV -Path C:\Foo\Users.Csv |
  Sort-Object  -Property Alias
$Users | Format-Table
```

3. Adding the users using the CSV

```
$Users |
  ForEach-Object -Parallel {
    $User = $_
    #  Create a hash table of properties to set on created user
    $Prop = @{}
    #  Fill in values
    $Prop.GivenName         = $User.Firstname
    $Prop.Initials          = $User.Initials
    $Prop.Surname           = $User.Lastname
    $Prop.UserPrincipalName = $User.UserPrincipalName + "@Reskit.Org"
    $Prop.Displayname       = $User.FirstName.Trim() + " " +
                              $User.LastName.Trim()
    $Prop.Description       = $User.Description
    $Prop.Name              = $User.Alias
    $PW = ConvertTo-SecureString -AsPlainText $User.Password -Force
    $Prop.AccountPassword   = $PW
    $Prop.ChangePasswordAtLogon = $true
    $Prop.Path                  = 'OU=IT,DC=Reskit,DC=ORG'
    $Prop.Enabled               = $true
    #  Now Create the User
    New-ADUser @Prop
    # Finally, Display User Created
    "Created $($Prop.Name)"
}
```

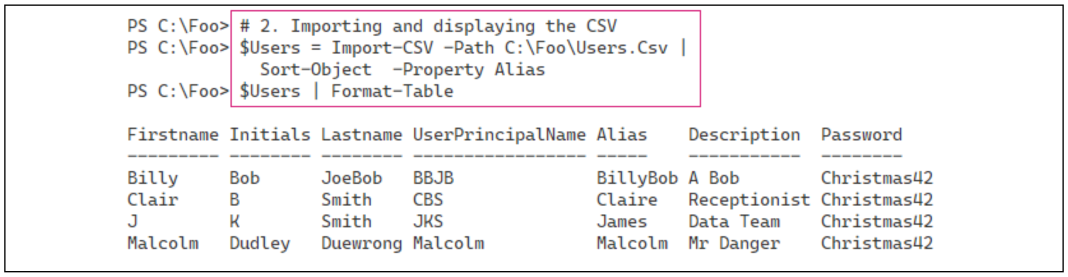
4. Showing all users in AD (`Reskit.Org`)

```
Get-ADUser -Filter * |
  Format-Table -Property Name, UserPrincipalName
```

How it works...

In *step 1*, which produces no output, you create a simple CSV file which you save to `C:\Foo\Users.CSV`.

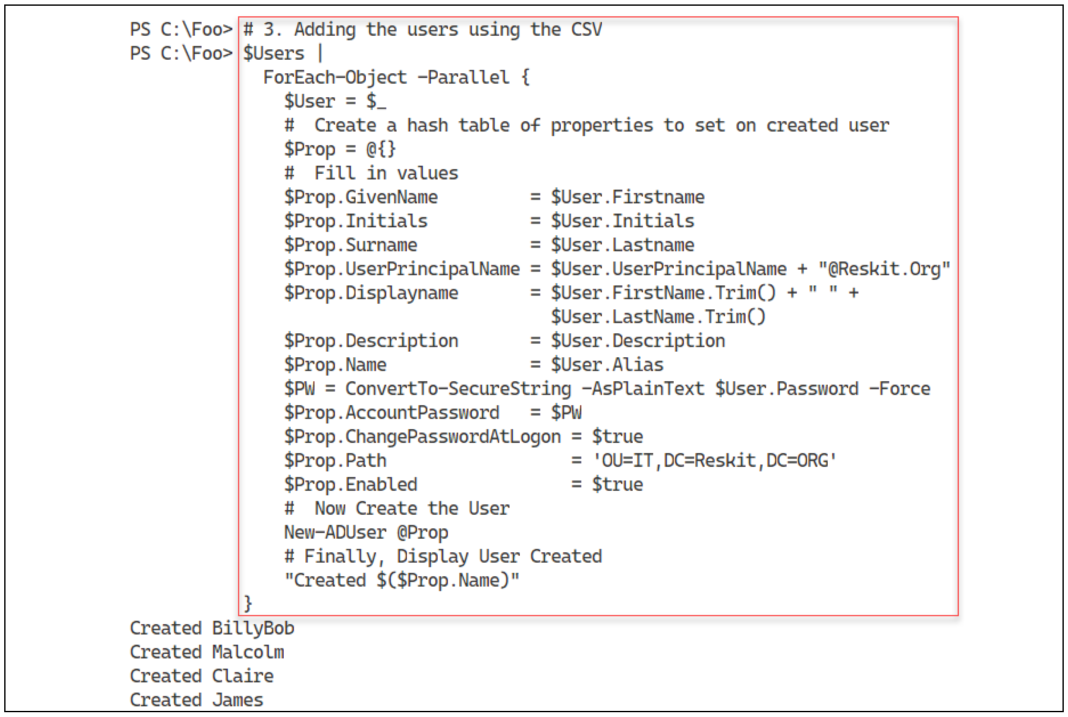
In *step 2*, you import this newly created CSV file and display the information it contains, which looks like this:



```
PS C:\Foo> # 2. Importing and displaying the CSV
PS C:\Foo> $Users = Import-CSV -Path C:\Foo\Users.Csv |
             Sort-Object  -Property Alias
PS C:\Foo> $Users | Format-Table

Firstname Initials Lastname UserPrincipalName Alias    Description  Password
--------- -------- -------- ----------------- -----    -----------  --------
Billy     Bob      JoeBob   BBJB              BillyBob A Bob        Christmas42
Clair     B        Smith    CBS               Claire   Receptionist Christmas42
J         K        Smith    JKS               James    Data Team    Christmas42
Malcolm   Dudley   Duewrong Malcolm           Malcolm  Mr Danger    Christmas42
```

Figure 6.36: Importing and displaying the CSV file

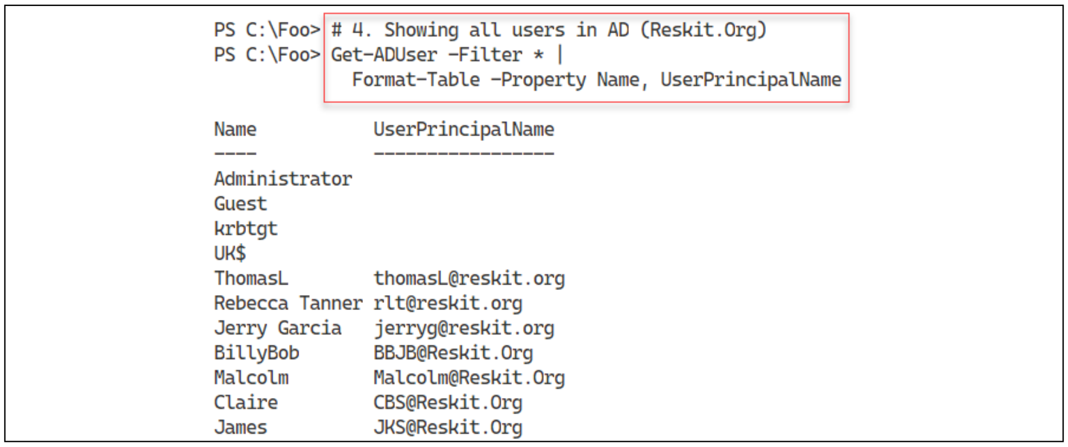
In *step 3*, you add each user contained in the CSV into AD. You add the users using `New-ADUser`, which itself produces no output. This step adds some output to show what users you added, which looks like this:



```
PS C:\Foo> # 3. Adding the users using the CSV
PS C:\Foo> $Users |
             ForEach-Object -Parallel {
               $User = $_
               #  Create a hash table of properties to set on created user
               $Prop = @{}
               #  Fill in values
               $Prop.GivenName         = $User.Firstname
               $Prop.Initials          = $User.Initials
               $Prop.Surname           = $User.Lastname
               $Prop.UserPrincipalName = $User.UserPrincipalName + "@Reskit.Org"
               $Prop.Displayname       = $User.FirstName.Trim() + " " +
                                         $User.LastName.Trim()
               $Prop.Description       = $User.Description
               $Prop.Name              = $User.Alias
               $PW = ConvertTo-SecureString -AsPlainText $User.Password -Force
               $Prop.AccountPassword   = $PW
               $Prop.ChangePasswordAtLogon = $true
               $Prop.Path                  = 'OU=IT,DC=Reskit,DC=ORG'
               $Prop.Enabled               = $true
               #  Now Create the User
               New-ADUser @Prop
               # Finally, Display User Created
               "Created $($Prop.Name)"
           }
Created BillyBob
Created Malcolm
Created Claire
Created James
```

Figure 6.37: Adding users into AD using the CSV file

In the final step in this recipe, *step 4*, you use `Get-ADUser` to view all the users in the `Reskit.Org` domain. This step's output looks like this:



```
PS C:\Foo> # 4. Showing all users in AD (Reskit.Org)
PS C:\Foo> Get-ADUser -Filter * |
             Format-Table -Property Name, UserPrincipalName

Name           UserPrincipalName
----           -----------------
Administrator
Guest
krbtgt
UK$
ThomasL        thomasL@reskit.org
Rebecca Tanner rlt@reskit.org
Jerry Garcia   jerryg@reskit.org
BillyBob       BBJB@Reskit.Org
Malcolm        Malcolm@Reskit.Org
Claire         CBS@Reskit.Org
James          JKS@Reskit.Org
```

Figure 6.38: Viewing all users in the Reskit.Org domain

There's more...

In *step 3*, you add users based on the CSV. You add these users explicitly to the IT OU, using the parameter `-Path` (as specified in the `$Prop` hash table). In production, when adding users that could reside in different OUs, you should extend the CSV to include the distinguished name of the OU into which you wish to add each user.

Creating Group Policy objects

A group policy allows you to define computer and user configuration settings that ensure a system remains configured per policy. Each time a domain-joined computer starts up and each time a domain user logs on, the local group policy agent on your computer obtains the group policy settings from AD and ensures they are applied.

In this recipe, you begin by first creating a GPO within the AD. You then configure the GPO, for example, enabling computers in the IT OU to use PowerShell scripts on those systems or set a specific screensaver. There are thousands of settings you can configure for a user or computer through a group policy. Microsoft has created a spreadsheet that lists the policy settings, which you can download from `https://www.microsoft.com/en-us/download/101451`. At the time of writing, the spreadsheet covers the Group Policy template files delivered with the Windows 10 May 2020 update (that is, Windows 10 2004).

Once you configure your GPO, you link the policy object to the OU you want to configure. You can also apply a GPO to the domain as a whole, to a specific AD site, or to an OU. You can also assign any GPO to multiple OUs, which can simplify your OU design.

The configuration of a GPO typically results in Windows generating information that a host's group policy agent (the code that applies the GPO objects) can access. This information tells the agent how to work. Settings made through administrative templates use registry settings inside `Registry.POL` files. The group policy agent obtains the policy details from the SYSVOL share on a DC and applies them whenever a user logs on or off or when a computer starts up or shuts down. The group policy module also provides the ability to create nice-looking reports describing the GPO.

Getting ready

You run this recipe on `DC1`, a DC in the `Reskit.Org` domain. You created this DC in *Installing an AD forest root domain* and after creating the IT OU. Also, ensure you have created the `C:\Foo` folder on `DC1` before continuing.

How to do it...

1. Creating a group policy object

```
$Pol = New-GPO -Name ITPolicy -Comment 'IT GPO' -Domain Reskit.Org
```

2. Ensuring just computer settings are enabled

```
$Pol.GpoStatus = 'UserSettingsDisabled'
```

3. Configuring the policy with two registry-based settings

```
$EPHT1= @{
  Name      = 'ITPolicy'
  Key       = 'HKLM\Software\Policies\Microsoft\Windows\PowerShell'
  ValueName = 'ExecutionPolicy'
  Value     = 'Unrestricted'
  Type      = 'String'
}
Set-GPRegistryValue @EPHT1 | Out-Null
$EPHT2= @{
  Name   = 'ITPolicy'
  Key    = 'HKLM\Software\Policies\Microsoft\Windows\PowerShell'
  ValueName = 'EnableScripts'
  Type   = 'DWord'
  Value  = 1
}
Set-GPRegistryValue @EPHT2 | Out-Null
```

4. Creating a screensaver GPO

```
$Pol2 = New-GPO -Name 'Screen Saver Time Out'
$Pol2.GpoStatus   = 'ComputerSettingsDisabled'
$Pol2.Description = '15 minute timeout'
```

5. Setting a Group Policy enforced registry value

```
$EPHT3= @{
  Name   = 'Screen Saver Time Out'
  Key    = 'HKCU\Software\Policies\Microsoft\Windows\' +
           'Control Panel\Desktop'
  ValueName = 'ScreenSaveTimeOut'
  Value  = 900
  Type   = 'DWord'
}
Set-GPRegistryValue @EPHT3 | Out-Null
```

6. Linking both GPOs to the IT OU

```
$GPLHT1 = @{
  Name     = 'ITPolicy'
  Target   = 'OU=IT,DC=Reskit,DC=org'
}
New-GPLink @GPLHT1 | Out-Null
$GPLHT2 = @{
  Name     = 'Screen Saver Time Out'
  Target   = 'OU=IT,DC=Reskit,DC=org'
}
New-GPLink @GPLHT2 | Out-Null
```

7. Displaying the GPOs in the domain

```
Get-GPO -All -Domain Reskit.Org |
  Sort-Object -Property DisplayName |
    Format-Table -Property Displayname, Description, GpoStatus
```

8. Creating and viewing a GPO report

```
$RPath = 'C:\Foo\GPOReport1.HTML'
Get-GPOReport -All -ReportType Html -Path $RPath
Invoke-Item -Path $RPath
```

9. Getting report in XML format

```
$RPath2 = 'C:\Foo\GPOReport2.XML'
Get-GPOReport -All -ReportType XML -Path $RPath2
$XML = [xml] (Get-Content -Path $RPath2)
```

10. Creating a simple GPO report

```
$FMTS = "{0,-33}  {1,-30} {2,-10} {3}"
$FMTS -f 'Name', 'Linked To', 'Enabled', 'No Override'
$FMTS -f '----', '---------', '-------', '-----------'
$XML.report.GPO |
  Sort-Object -Property Name |
    ForEach-Object {
     $Gname = $_.Name
     $SOM = $_.linksto.SomPath
     $ENA = $_.linksto.enabled
     $NOO = $_.linksto.nooverride
     $FMTS -f $Gname, $SOM, $ENA, $NOO
   }
```

How it works...

Note that, like many AD-related cmdlets, the cmdlets you use to manage GPOs do not produce much output.

In *step 1*, you create a new GPO in the Reskit.Org domain. This step creates an empty GPO. This GPO is not yet linked to any OU and thus does not get applied.

In *step 2*, you disable user settings, which allows the GPO client to ignore any user settings. Doing so can make the client GPO processing a bit faster.

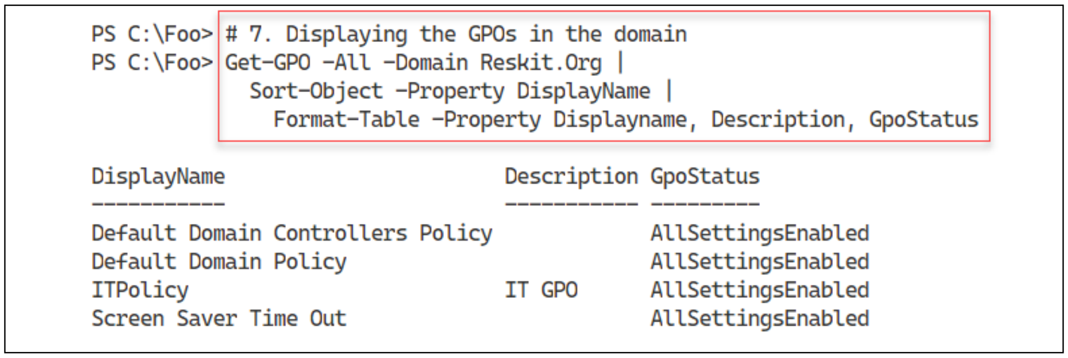
In *step 3*, you set this GPO to have two specific registry-based values. When a computer starts up, the GPO processing on that client computer ensures that these two registry values are set on the client. During Group Policy refresh (which happens approximately every 2 hours) the group policy agent enforces the value in the policy.

In *step 4* and *step 5*, you create a new GPO and set a screen saver timeout of 900 seconds.

In *step 6*, you link the two GPOs to the IT OU. Until you link the GPOs to an OU (or to the domain or a domain site), GPO processing ignores the GPO.

In this recipe, *step 1* through *step 6* produce no output.

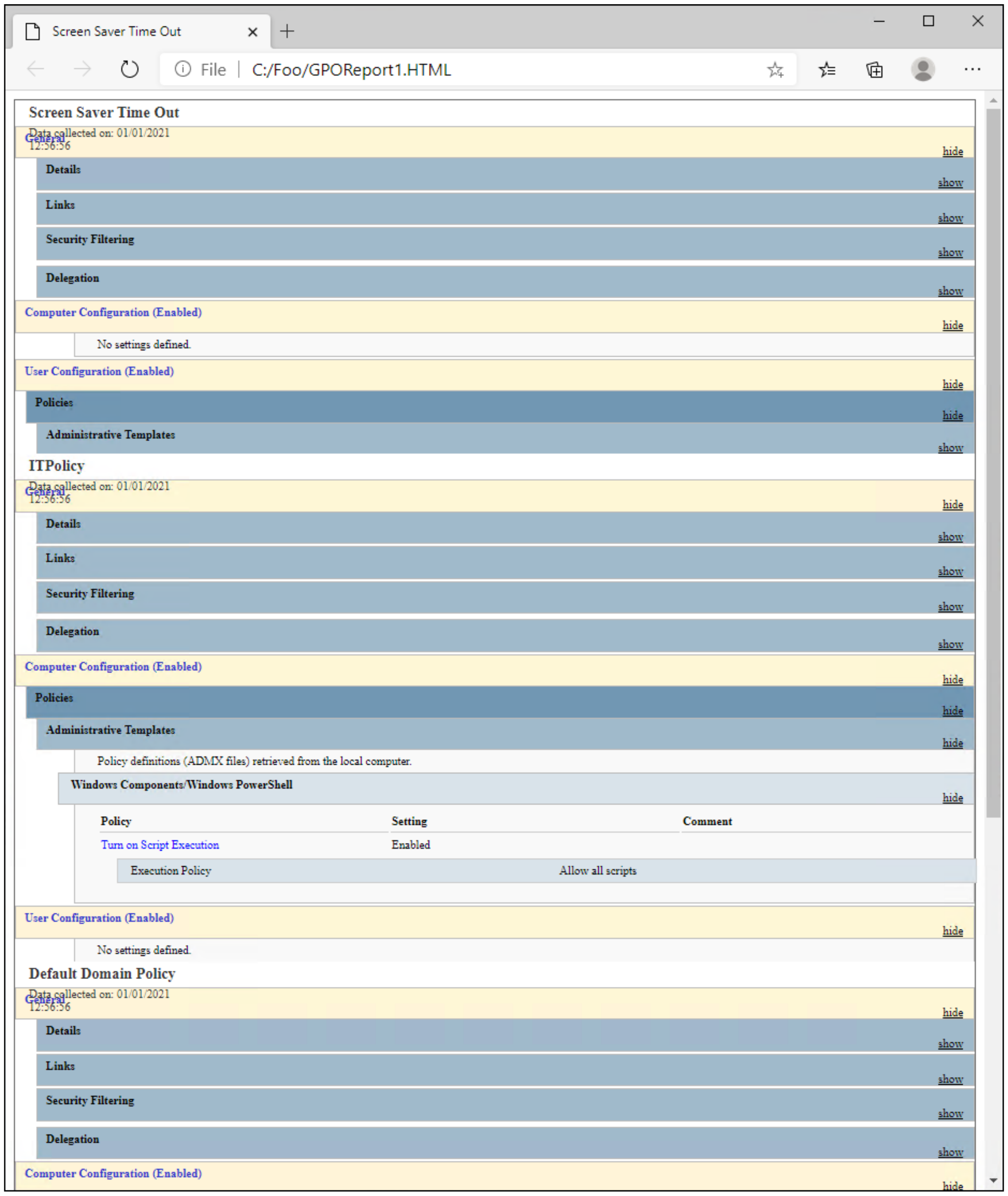
In *step 7*, you use `Get-GPO` to return information about all the GPOs in the domain, which looks like this:



```
PS C:\Foo> # 7. Displaying the GPOs in the domain
PS C:\Foo> Get-GPO -All -Domain Reskit.Org |
             Sort-Object -Property DisplayName |
               Format-Table -Property Displayname, Description, GpoStatus

DisplayName                       Description GpoStatus
-----------                       ----------- ---------
Default Domain Controllers Policy             AllSettingsEnabled
Default Domain Policy                         AllSettingsEnabled
ITPolicy                          IT GPO      AllSettingsEnabled
Screen Saver Time Out                         AllSettingsEnabled
```

Figure 6.39: Displaying all the GPOs in the domain

In *step 8*, you generate and display a GPO report by using the `Get-GPOReport` command. The command produces no output, but by using `Invoke-Command`, you view the report in your default browser, which looks like this:



Screen Saver Time Out

File | C:/Foo/GPOReport1.HTML

Screen Saver Time Out

Data collected on: 01/01/2021 12:56:56

General — hide

- **Details** — show
- **Links** — show
- **Security Filtering** — show
- **Delegation** — show

Computer Configuration (Enabled) — hide

No settings defined.

User Configuration (Enabled) — hide

- **Policies** — hide
  - **Administrative Templates** — show

ITPolicy

Data collected on: 01/01/2021 12:56:56

General — hide

- **Details** — show
- **Links** — show
- **Security Filtering** — show
- **Delegation** — show

Computer Configuration (Enabled) — hide

- **Policies** — hide
  - **Administrative Templates** — hide

Policy definitions (ADMX files) retrieved from the local computer.

Windows Components/Windows PowerShell — hide

Policy	Setting	Comment
Turn on Script Execution	Enabled	
Execution Policy	Allow all scripts	

User Configuration (Enabled) — hide

No settings defined.

Default Domain Policy

Data collected on: 01/01/2021 12:56:56

General — hide

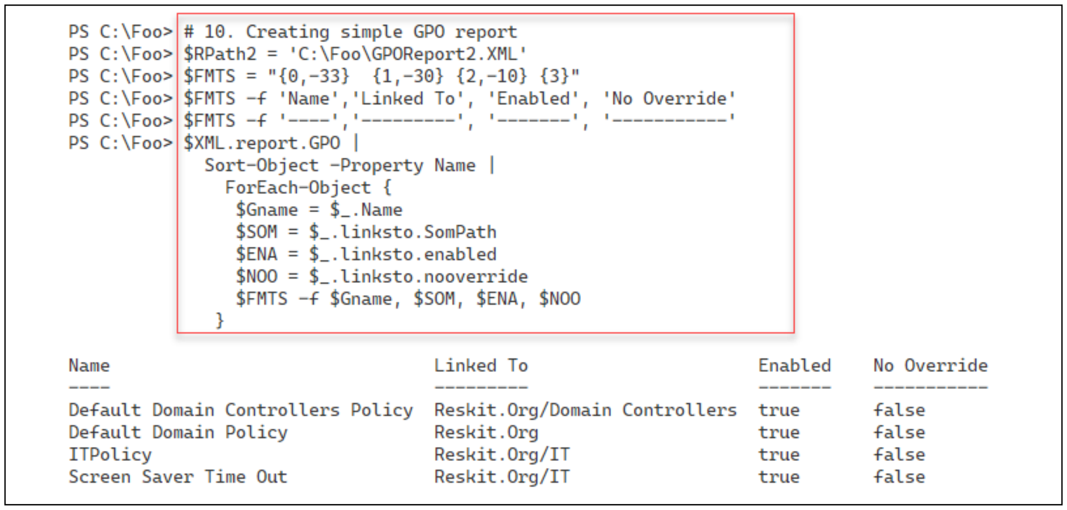
- **Details** — show
- **Links** — show
- **Security Filtering** — show
- **Delegation** — show

Computer Configuration (Enabled) — hide

Figure 6.40: Viewing the GPO report in browser

In *step 9*, you use `Get-GPOReport` to return a report of all the GPOs in the domain in an XML format, which produces no output.

In *step 10*, you iterate through the returned XML and produce a simple report on GPOs and where they are linked, which looks like this:



```
PS C:\Foo> # 10. Creating simple GPO report
PS C:\Foo> $RPath2 = 'C:\Foo\GPOReport2.XML'
PS C:\Foo> $FMTS = "{0,-33}  {1,-30} {2,-10} {3}"
PS C:\Foo> $FMTS -f 'Name','Linked To', 'Enabled', 'No Override'
PS C:\Foo> $FMTS -f '----','---------', '-------', '-----------'
PS C:\Foo> $XML.report.GPO |
             Sort-Object -Property Name |
               ForEach-Object {
                $Gname = $_.Name
                $SOM = $_.linksto.SomPath
                $ENA = $_.linksto.enabled
                $NOO = $_.linksto.nooverride
                $FMTS -f $Gname, $SOM, $ENA, $NOO
              }

Name                              Linked To                      Enabled    No Override
----                              ---------                      -------    -----------
Default Domain Controllers Policy Reskit.Org/Domain Controllers true       false
Default Domain Policy             Reskit.Org                     true       false
ITPolicy                          Reskit.Org/IT                  true       false
Screen Saver Time Out             Reskit.Org/IT                  true       false
```

Figure 6.41: Creating a simple report on GPOs

There's more...

In *step 8*, you see the output from the `Get-GPOReport` cmdlet. At the time of writing, the Edge browser does not render the output as nicely as you might want. What may look like a bug in the graphic is just how Edge (again, at the time of writing) renders this HTML document.

In *step 9* and *step 10*, you create your mini-report. In *step 9*, you use the `Get-GPOReport` command to obtain a report of all GPOs in the domain returned as XML. In *step 10*, you report on the GPOs in the Reskit domain using .NET composite string formatting with the `-f` operator.

Using .NET composite formatting enables you to create nice-looking output when the objects returned by a cmdlet are not in a form to be used directly with `Format-Table`. In *step 9*, for example, the XML returned contains details of the GPO links as a property which is actually an object with sub-properties. To create nice-looking output, you create a format string you use to display each row of your report, including report header lines. You first use this format string to create the two report header lines for the report. Then, for each GPO in the returned list, you obtain the GPO name, which OU the GPO is linked to, whether it is enabled, and whether the GPO is non-overridable. Finally, you use the format string to output a single report line. In most cases, these custom reports are easier to read and contain only the information you deem useful.

As an alternative to creating a report as shown in *step 9*, you could have created a custom hash table and used it to create a customized object for your report.

The `Format-Table` and `Format-List` cmdlets are of most use when each object has simple properties. When an object has a property that is an object with properties, the format commands do not surface these (sub)properties. In this case, you must obtain the details manually of the GPO links for each GPO and generate your report lines based on those results. While the report layout works well for this specific set of GPOs, should you have GPOs with longer names, or linked to deeper OUs, you may need to adjust the format string you set in *step 10*.

Reporting on AD replication

AD uses a special database to support its operations. The database is a distributed, multi-master database with convergence – every DC in every domain stores this database in the file `C:\Windows\NTDS\ntds.dit`.

Every DC in any domain holds a complete copy of this database. If you add a new user or change a user's office, that change occurs on just one DC (initially). AD replication makes the change in all database copies. In this way, the database remains consistent over time and across all DCs.

AD replication is based on partitions – a slice of the overall database. AD can replicate each partition separately. There are several partitions in AD:

- **Schema partition**: This holds the AD schema that defines each object that AD stores in the database. The schema also defines the properties of all these objects.
- **Configuration partition**: This holds the details of the structure of the domain.
- **Domain partition**: This partition, also known as the domain naming context, contains the objects relating to a domain (users, groups, OUs, and so on). The objects in this partition are defined based on the schema.
- **Application partition**: Some applications, such as DNS, store objects in your AD and rely on AD replication to replicate the values.

There are two types of replication: intra-site replication and inter-site replication. Intra-site replication happens between DCs in a given AD site, while inter-site replication occurs between different sites.

You can create different topologies for replication, including:

- **Ring**: Each DC in a site has at least two inbound replication partners. When any change is made to any DC, that DC notifies its replication partners that it has a change. Those DCs can then replicate that change (if they have not seen the change before). AD by default ensures there are no more than three hops within the replication topology. If you have a large number of DCs (more than seven), AD automatically creates additional replication links to keep the hop count below three.
- **Full mesh**: With this topology, all DCs replicate to all others. Full mesh replication keeps the database in sync with a minimum of replication delay, but can be more expensive in terms of bandwidth (and DC utilization). It is not scalable.
- **Hub and spoke**: You might use this approach in enormous organizations where a "spoke" DC replicates with a central hub DC. The hub DC then replicates the change to all other spoke DCs in your organization. Hub and spoke can reduce replication for widely dispersed implementations.
- **Hybrid**: Here you can combine any of the above, based on business needs.

By default, AD replication uses a ring topology. You can adopt different topologies if your business needs dictate, but this requires configuration.

In most smaller organizations (such as `Reskit.Org`), replication is set up and operates automagically. But for large and distributed organizations, replication can be quite complicated as you attempt to balance being totally up to date against the cost of geographical bandwidth. If you have DCs in several continents, you want to collect those changes and do them all at once, say every 4 hours. But that means the remote DC would have out-of-date information for a period. As a general rule of thumb, you should design replication so that it happens faster than a person can fly between your AD sites served by a DC. If you change your password, say in London, then as long as the changes occur within 12 hours, when you fly to, say Brisbane, Australia, the Australian DCs will contain the replicated password. Thus, you can log in with the new password immediately upon landing in Brisbane.

For more information on AD replication concepts, see `https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/replication/active-directory-replication-concepts`.

For any sizeable organization, the design and planning of AD is vital – see `https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/ad-ds-design-and-planning` for more information on the planning and design work necessary.

Traditionally, you use many Win32 console applications to manage and troubleshoot replication, including `repadmin.exe`, which broadly replaces an earlier command `replmon.exe`. For some detail on the `repadmin.exe` command (and replication) see `https://techcommunity.microsoft.com/t5/ask-the-directory-services-team/getting-over-replmon/ba-p/396687`.

With the advent of PowerShell and the PowerShell AD module, you can now perform many of the functions of `repadmin.exe` using PowerShell cmdlets. In this recipe, you examine some of the details of AD replication using PowerShell.

Getting ready

You run this recipe on `DC1`, a DC in the Reskit.Org domain. You should also have `DC2` and `UKDC1` available and online at the start of testing this recipe. You must have installed PowerShell 7 and VS Code on all these hosts.

How to do it...

1. Checking replication partners for `DC1`

```
Get-ADReplicationPartnerMetadata -Target DC1.Reskit.Org   |
  Format-List -Property Server, PartnerType, Partner,
               Partition, LastRep*
```

2. Checking AD replication partner metadata in the domain

```
Get-ADReplicationPartnerMetadata -Target Reskit.Org -Scope Domain |
  Format-Table -Property Server, P*Type,Last*
```

3. Investigating group membership metadata

```
$REPLHT = @{
  Object              = (Get-ADGroup -Identity 'IT Team')
  Attribute           = 'Member'
  ShowAllLinkedValues = $true
  Server              = (Get-ADDomainController)
}
Get-ADReplicationAttributeMetadata @REPLHT |
  Format-Table -Property A*NAME, A*VALUE, *TIME
```

4. Adding two users to the group and removing one

```
Add-ADGroupMember -Identity 'IT Team' -members Malcolm
Add-ADGroupMember -Identity 'IT Team' -members Claire
Remove-ADGroupMember -Identity 'IT Team' -members Claire -Confirm:$False
```

5. Checking updated metadata

```
Get-ADReplicationAttributeMetadata @REPLHT |
  Format-Table -Property A*NAME, A*VALUE, *TIME
```

6. Creating an initial replication failure report

```
$DomainController = 'DC1'
$Report = [ordered] @{}
## Replication Partners ##
$ReplMeta =
    Get-ADReplicationPartnerMetadata -Target $DomainController
$Report.ReplicationPartners = $ReplMeta.Partner
$Report.LastReplication     = $ReplMeta.LastReplicationSuccess
## Replication Failures ##
$REPLF = Get-ADReplicationFailure -Target $DomainController
$Report.FailureCount  = $REPLF.FailureCount
$Report.FailureType   = $REPLF.FailureType
$Report.FirstFailure  = $REPLF.FirstFailureTime
$Report.LastFailure   = $REPLF.LastFailure
$Report
```

7. Simulating a connection issue

```
Stop-Computer DC2  -Force
Start-Sleep -Seconds 30
```

8. Making a change to this AD

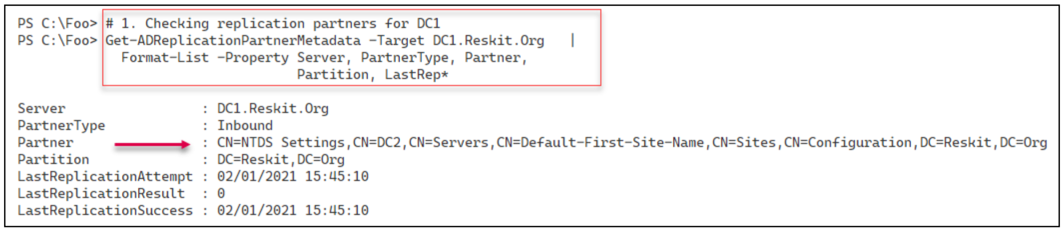
```
Get-AdUser -identity BillyBob  |
  Set-AdUser -Office 'Cookham Office' -Server DC1
```

9. Using `repadmin.exe` to generate a status report

```
repadmin /replsummary
```

How it works...

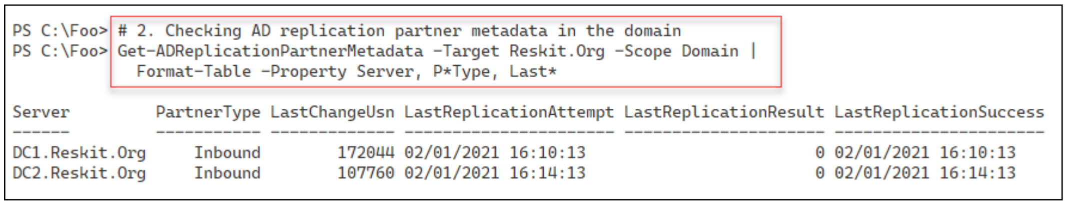
In *step 1*, you check to discover replication partners for `DC1`. With two DCs in the Reskit domain (`DC1` and `DC2`), the output of this step looks like this:



```
PS C:\Foo> # 1. Checking replication partners for DC1
PS C:\Foo> Get-ADReplicationPartnerMetadata -Target DC1.Reskit.Org   |
             Format-List -Property Server, PartnerType, Partner,
                                   Partition, LastRep*

Server                    : DC1.Reskit.Org
PartnerType               : Inbound
Partner                   : CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=Reskit,DC=Org
Partition                 : DC=Reskit,DC=Org
LastReplicationAttempt    : 02/01/2021 15:45:10
LastReplicationResult     : 0
LastReplicationSuccess    : 02/01/2021 15:45:10
```

Figure 6.42: Checking replication partners for DC1

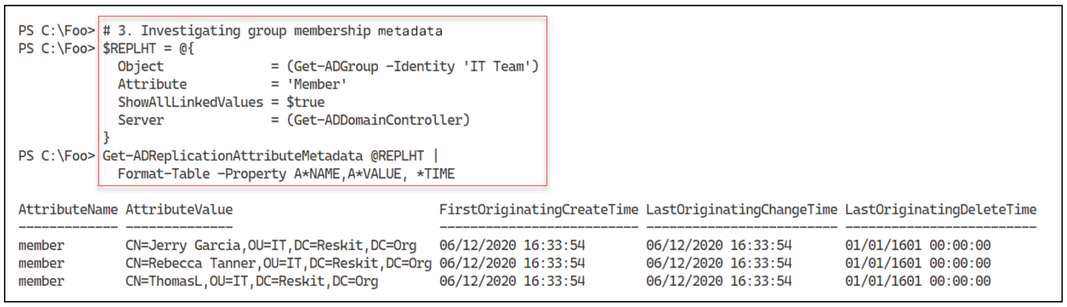
In *step 2*, you check on the replication partner metadata for the entire domain, which looks like this:



```
PS C:\Foo> # 2. Checking AD replication partner metadata in the domain
PS C:\Foo> Get-ADReplicationPartnerMetadata -Target Reskit.Org -Scope Domain |
             Format-Table -Property Server, P*Type, Last*

Server          PartnerType LastChangeUsn LastReplicationAttempt LastReplicationResult LastReplicationSuccess
------          ----------- ------------- ---------------------- --------------------- ----------------------
DC1.Reskit.Org     Inbound         172044 02/01/2021 16:10:13                        0 02/01/2021 16:10:13
DC2.Reskit.Org     Inbound         107760 02/01/2021 16:14:13                        0 02/01/2021 16:14:13
```

Figure 6.43: Checking replication partner metadata in the domain

In *step 3*, you examine the metadata for group membership. You take this step after creating the `IT Team` security group and populating it, at which point the output looks like this:

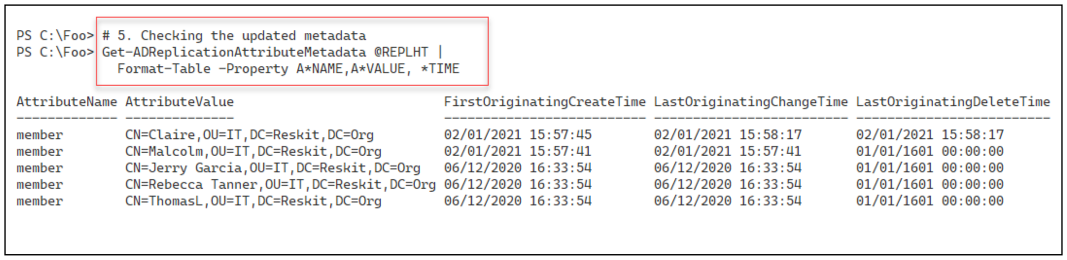


```
PS C:\Foo> # 3. Investigating group membership metadata
PS C:\Foo> $REPLHT = @{
             Object             = (Get-ADGroup -Identity 'IT Team')
             Attribute          = 'Member'
             ShowAllLinkedValues = $true
             Server             = (Get-ADDomainController)
           }
PS C:\Foo> Get-ADReplicationAttributeMetadata @REPLHT |
             Format-Table -Property A*NAME,A*VALUE, *TIME

AttributeName AttributeValue                           FirstOriginatingCreateTime LastOriginatingChangeTime LastOriginatingDeleteTime
------------- --------------                           -------------------------- ------------------------- -------------------------
member        CN=Jerry Garcia,OU=IT,DC=Reskit,DC=Org   06/12/2020 16:33:54        06/12/2020 16:33:54       01/01/1601 00:00:00
member        CN=Rebecca Tanner,OU=IT,DC=Reskit,DC=Org 06/12/2020 16:33:54        06/12/2020 16:33:54       01/01/1601 00:00:00
member        CN=ThomasL,OU=IT,DC=Reskit,DC=Org        06/12/2020 16:33:54        06/12/2020 16:33:54       01/01/1601 00:00:00
```

Figure 6.44: Examining group membership metadata

In *step 4*, which creates no output, you change this group by adding two members. Then, you remove one of them from the group. Under the covers, this step updates the group membership (three times), which generates replication traffic to replicate the group membership changes from `DC1` to `DC2`.

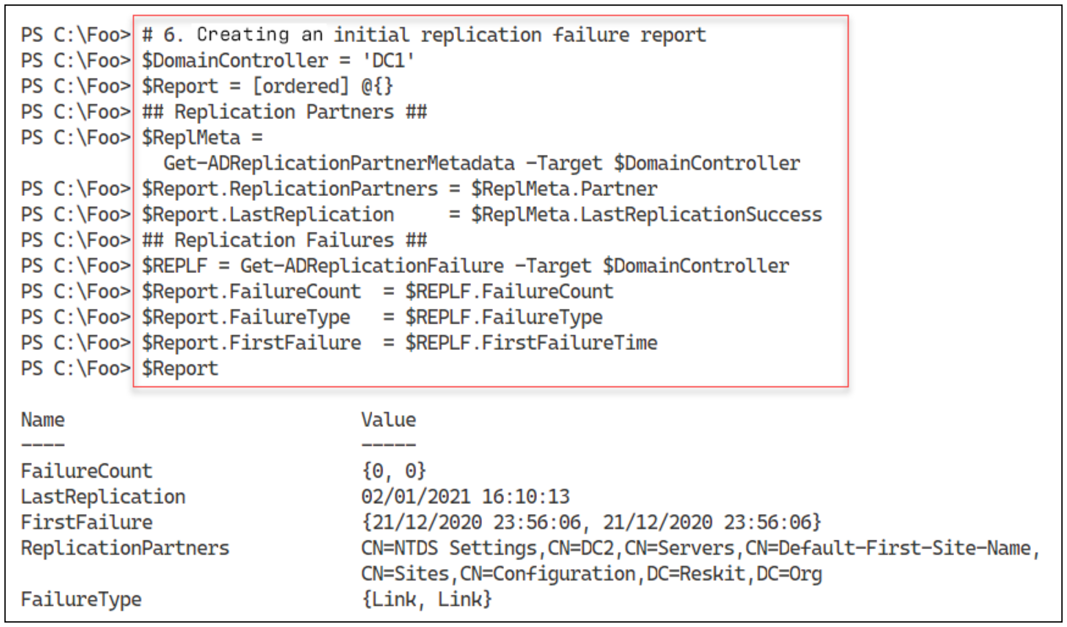
With *step 5*, you re-examine the group membership metadata to see the effects of *step 4*, which looks like this:



```
PS C:\Foo> # 5. Checking the updated metadata
PS C:\Foo> Get-ADReplicationAttributeMetadata @REPLHT |
             Format-Table -Property A*NAME,A*VALUE, *TIME

AttributeName AttributeValue                           FirstOriginatingCreateTime LastOriginatingChangeTime LastOriginatingDeleteTime
------------- --------------                           -------------------------- ------------------------- -------------------------
member        CN=Claire,OU=IT,DC=Reskit,DC=Org         02/01/2021 15:57:45        02/01/2021 15:58:17       02/01/2021 15:58:17
member        CN=Malcolm,OU=IT,DC=Reskit,DC=Org        02/01/2021 15:57:41        02/01/2021 15:57:41       01/01/1601 00:00:00
member        CN=Jerry Garcia,OU=IT,DC=Reskit,DC=Org   06/12/2020 16:33:54        06/12/2020 16:33:54       01/01/1601 00:00:00
member        CN=Rebecca Tanner,OU=IT,DC=Reskit,DC=Org 06/12/2020 16:33:54        06/12/2020 16:33:54       01/01/1601 00:00:00
member        CN=ThomasL,OU=IT,DC=Reskit,DC=Org        06/12/2020 16:33:54        06/12/2020 16:33:54       01/01/1601 00:00:00
```

Figure 6.45: Checking the updated metadata

In *step 6*, you create a report of replication failures, with output like this:

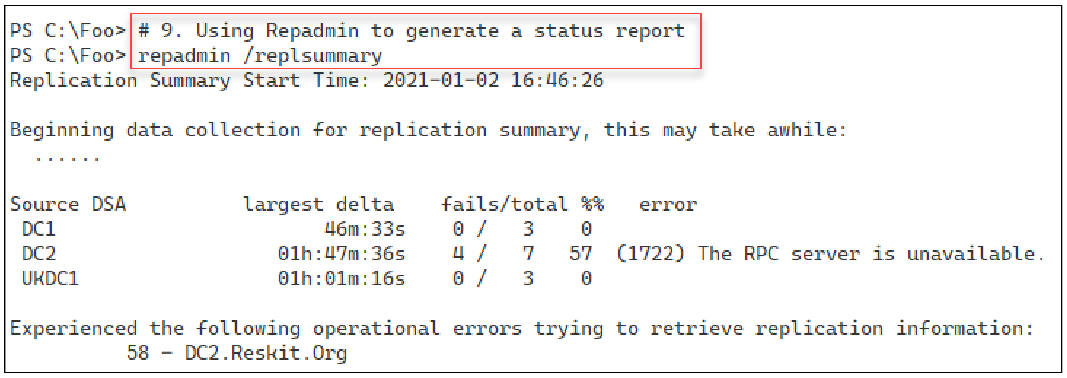


```
PS C:\Foo> # 6. Creating an initial replication failure report
PS C:\Foo> $DomainController = 'DC1'
PS C:\Foo> $Report = [ordered] @{}
PS C:\Foo> ## Replication Partners ##
PS C:\Foo> $ReplMeta =
             Get-ADReplicationPartnerMetadata -Target $DomainController
PS C:\Foo> $Report.ReplicationPartners = $ReplMeta.Partner
PS C:\Foo> $Report.LastReplication     = $ReplMeta.LastReplicationSuccess
PS C:\Foo> ## Replication Failures ##
PS C:\Foo> $REPLF = Get-ADReplicationFailure -Target $DomainController
PS C:\Foo> $Report.FailureCount  = $REPLF.FailureCount
PS C:\Foo> $Report.FailureType   = $REPLF.FailureType
PS C:\Foo> $Report.FirstFailure  = $REPLF.FirstFailureTime
PS C:\Foo> $Report

Name                           Value
----                           -----
FailureCount                   {0, 0}
LastReplication                02/01/2021 16:10:13
FirstFailure                   {21/12/2020 23:56:06, 21/12/2020 23:56:06}
ReplicationPartners            CN=NTDS Settings,CN=DC2,CN=Servers,CN=Default-First-Site-Name,
                               CN=Sites,CN=Configuration,DC=Reskit,DC=Org
FailureType                    {Link, Link}
```

Figure 6.46: Creating a report of replication failures

In *step 7*, you simulate a connection issue by stopping `DC2` and waiting until the system has completed its shutdown. In *step 8*, you make a change to a user (on `DC1`). These steps create no output, although *step 7* results in AD attempting to replicate the changed user account to `DC2`.

In *step 9*, you use the `repadmin.exe` command to generate a replication summary report, which looks like this:



```
PS C:\Foo> # 9. Using Repadmin to generate a status report
PS C:\Foo> repadmin /replsummary
Replication Summary Start Time: 2021-01-02 16:46:26

Beginning data collection for replication summary, this may take awhile:
  ......

Source DSA          largest delta    fails/total %%   error
 DC1                      46m:33s    0 /   3    0
 DC2                  01h:47m:36s    4 /   7   57  (1722) The RPC server is unavailable.
 UKDC1                01h:01m:16s    0 /   3    0

Experienced the following operational errors trying to retrieve replication information:
          58 - DC2.Reskit.Org
```

Figure 6.47: Generating a replication status report using repadmin

There's more...

In *step 2*, you discover the replication partners within the domain. In a domain with just two DCs, you would expect each DC to be an inbound replication partner to the other DC, and that is what the figure shows.

In *step 3* through *step 5*, you examine the change in replication data before and after a group membership change. Using the `Get-ADReplicationAttributeMetadata` cmdlet is an excellent way to discover the specific changes to a security group's membership, including removing a member. The information provided does not tell you who made the change or the system where the update originated. But knowing that someone made a change and when are essential first steps.

In *step 7*, you create a potential replication issue by stopping `DC2`. In *step 8*, you generate a change to the AD on `DC1`, which would typically be replicated very quickly to `DC2`. But since `DC2` is down, that replication cannot happen.

In *step 9*, you use the `repadmin.exe` console application to generate a replication summary showing that `DC1` cannot contact `DC2`. Sometimes, older tools, like `repadmin.exe`, produce better output than the cmdlets. For console usage and when you want a simple summary of replication, `repadmin.exe` is still a great tool.

Reporting on AD computers

Monitoring the AD is a necessary albeit time-consuming task. With larger numbers of users and computers to manage, you need all the help you can get, and PowerShell makes it easy to keep track of things.

A computer that has not logged on for an extended period could represent a security risk or could be a lost/stolen computer. It could also be a system that you have not rebooted after having applied patches and updates.

This recipe creates a report of computers that have not logged on or that you have not rebooted for a while.

One challenge in developing scripts like this is creating meaningful test data. If you wish to generate a test report showing a system that has not logged in for over 6 months, you might have to wait for 6 months to get the necessary data. This recipe shows a way around that for testing purposes.

Getting ready

You run this recipe on `DC1`, a DC in the Reskit domain on which you have installed PowerShell 7 and VS Code. Also, you should have completed earlier recipes that created some computer objects inside the AD, including *Creating and managing AD users and groups*.

How to do it...

1. Creating example computer accounts in the AD

```
$NCHT1 = @{
    Name        = 'NLIComputer1_1week'
    Description = 'Computer last logged in 1 week ago'
}
New-ADComputer @NCHT1
$NCHT2 = @{
  Name        = 'NLIComputer2_1month'
  Description = 'Computer last logged in 1 week ago'
}
New-ADComputer @NCHT2
$NCHT3 = @{
  Name        = 'NLIComputer3_6month'
  Description = 'Computer last logged in 1 week ago'
}
New-ADComputer @NCHT3
```

2. Creating some constants for later comparison

```
$OneWeekAgo   = (Get-Date).AddDays(-7)
$OneMonthAgo  = (Get-Date).AddMonths(-1)
$SixMonthsAgo = (Get-Date).AddMonths(-6)
```

3. Defining a function to create sample data

```
Function Get-RKComputers {
$ADComputers = Get-ADComputer -Filter * -Properties LastLogonDate
$Computers = @()
foreach ($ADComputer in $ADComputers) {
  $Name = $ADComputer.Name
  # Real computers and last logon date
  if ($adComputer.name -NotMatch '^NLI') {
    $LLD = $ADComputer.LastLogonDate
  }
  Elseif ($ADComputer.Name -eq 'NLIComputer1_1week')  {
    $LLD = $OneWeekAgo.AddMinutes(-30)
  }
  Elseif ($ADComputer.Name -eq 'NLIComputer2_1month')  {
    $LLD = $OneMonthAgo.AddMinutes(-30)
  }
  Elseif ($ADComputer.Name -eq 'NLIComputer3_6month')  {
    $LLD = $SixMonthsAgo.AddMinutes(-30)
  }
  $Computers += [pscustomobject] @{
    Name = $Name
    LastLogonDate = $LLD
  }
 }
 $Computers
}
```

4. Building the report header

```
$RKReport = ''           # Start of report
$RKReport += "*** Reskit.Org AD Daily AD Computer Report`n"
$RKReport += "*** Generated [$(Get-Date)]`n"
$RKReport += "***********************************`n`n"
```

5. Getting computers in RK AD using `Get-RKComputers`

```
$Computers = Get-RKComputers
```

6. Getting computers that have never logged on

```
$RKReport += "Computers that have never logged on`n"
$RkReport += "Name                    LastLogonDate`n"
$RkReport += "----                    -------------`n"
$RKReport += Foreach($Computer in $Computers) {
  If ($null -eq $Computer.LastLogonDate) {
   "{0,-22}  {1}  `n" -f $Computer.Name, "Never"
  }
}
```

7. Reporting on computers that have not logged on in over 6 months

```
$RKReport += "`nComputers that havent logged in over 6 months`n"
$RkReport += "Name                    LastLogonDate`n"
$RkReport += "----                    -------------`n"
$RKReport +=
foreach($Computer in $Computers) {
  If (($Computer.LastLogonDate -lt $SixMonthsAgo) -and
      ($null -ne $Computer.LastLogonDate)) {
 ("`n{0,-23}  {1}  `n"
-f $Computer.Name, $Computer.LastLogonDate).trim()
  }
}
```

8. Reporting on computer accounts that have not logged in for 1–6 months

```
$RKReport += "`n`nComputers that havent logged in 1-6 months`n"
$RkReport += "Name                    LastLogonDate`n"
$RkReport += "----                    -------------"
$RKReport +=
foreach($Computer in $Computers) {
  If (($Computer.LastLogonDate -ge $SixMonthsAgo) -and
     ($Computer.LastLogonDate -lt $OneMonthAgo) -and
       ($null -ne $Computer.LastLogonDate)) {
   "`n{0,-22}  {1}  " -f $Computer.Name, $Computer.LastLogonDate
  }
}
```

9. Reporting on computer accounts that have not logged in
   for the past 1 week to 1 month

```
$RKReport += "`n`nComputers that have between one week "
$RKReport += "and one month ago`n"
$RkReport += "Name                    LastLogonDate`n"
$RkReport += "----                    ------------"
$RKReport +=
foreach($Computer in $Computers) {
  If (($Computer.LastLogonDate -ge $OneMonthAgo) -and
     ($Computer.LastLogonDate -lt $OneWeekAgo) -and
       ($null -ne $Computer.LastLogonDate)) {
   "`n{0,-22}  {1}  " -f $Computer.Name, $Computer.LastLogonDate
  }
}
```

10. Displaying the report

```
$RKReport
```

How it works...

In this recipe, all but the final step produce no output. Some of the steps exist to enable you to test the report that this recipe generates. Some of these steps might not be necessary for real life, if you already have enough real-life data to create a complete report.

In *step 1*, you create three AD computer accounts. You use these accounts to simulate computers that have not logged on for a while, thus enabling you to view a complete report. If you re-run this recipe or this step, adding these accounts produces errors since the accounts already exist. You could modify this step to check to see if the accounts exist before creating them.

In *step 2*, you create three time constants, representing the time 7 days ago, 1 month ago, and 6 months ago. This step enables you to test if a given user account has not logged on in that period.

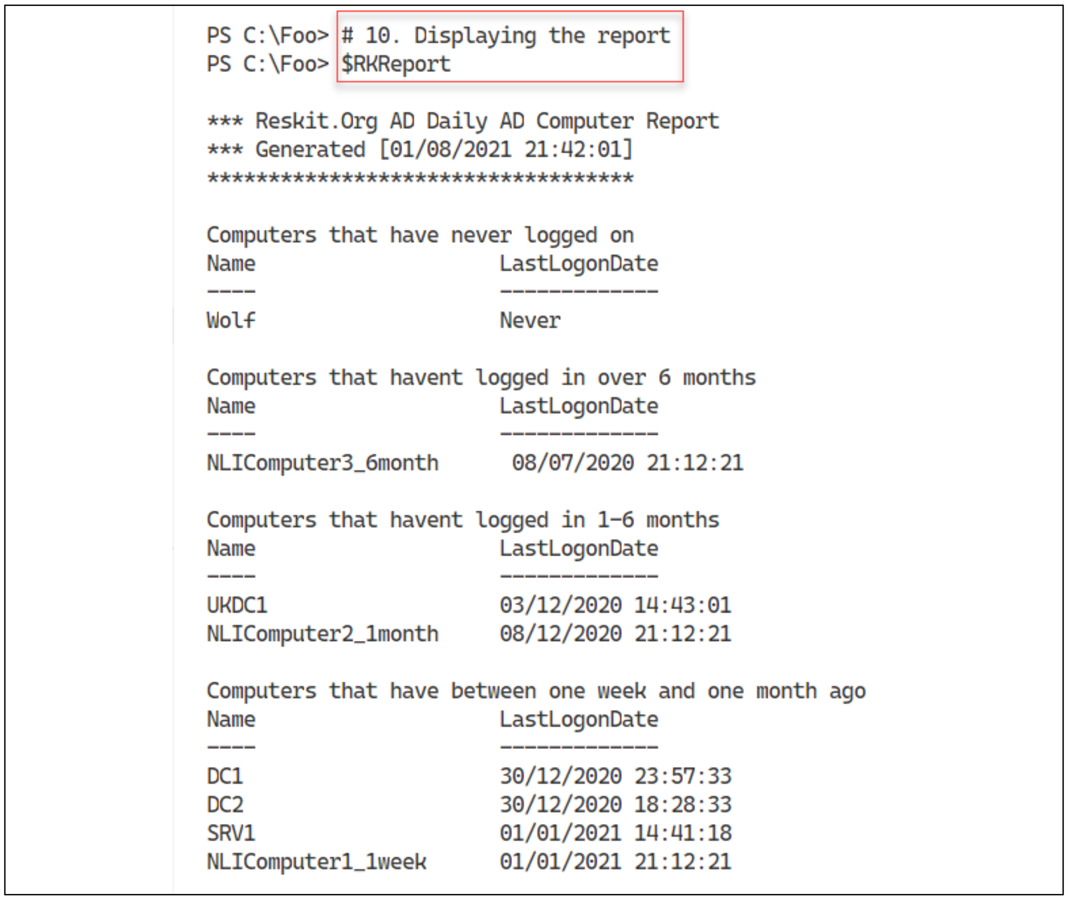
In *step 3*, you create a new function, `Get-RKComputers`. This function returns a list of all the computer accounts in AD along with their last logon time.

In *step 4*, you begin the report by creating a report header.

In *step 5*, you call the `Get-RKComputers` and populate the `$Computers` array (all the computers available).

In *step 6* through *step 9*, you add details to the report of computers that have never logged on, have not logged on in over 6 months, have not logged on in 1–6 months, and computers that have not logged on in 1 week to 1 month.

In the final step, *step 10*, you display the report created by the earlier steps. The output of this step looks like this:



```
PS C:\Foo> # 10. Displaying the report
PS C:\Foo> $RKReport

*** Reskit.Org AD Daily AD Computer Report
*** Generated [01/08/2021 21:42:01]
***********************************

Computers that have never logged on
Name                    LastLogonDate
----                    -------------
Wolf                    Never

Computers that havent logged in over 6 months
Name                    LastLogonDate
----                    -------------
NLIComputer3_6month      08/07/2020 21:12:21

Computers that havent logged in 1-6 months
Name                    LastLogonDate
----                    -------------
UKDC1                   03/12/2020 14:43:01
NLIComputer2_1month     08/12/2020 21:12:21

Computers that have between one week and one month ago
Name                    LastLogonDate
----                    -------------
DC1                     30/12/2020 23:57:33
DC2                     30/12/2020 18:28:33
SRV1                    01/01/2021 14:41:18
NLIComputer1_1week      01/01/2021 21:12:21
```

Figure 6.48: Displaying the report

There's more...

In *step 3*, you create a function to get the computer accounts in AD. This function returns an array of computer names and the last logon date. In production, you might amend this function to return computer accounts from just certain OUs. You could also extend this function to test whether each computer is online by testing a network connection to the computer, or checking to see if there is a DNS A record for the computer to detect stale computer accounts.

In *step 5* through *step 9*, you create a report by adding text lines to the `$RKReport` variable. In doing so, you need to add CRLF characters before or after each line of text when you add each line to the report. Ensuring each line of the report begins in the right place can be challenging in creating reports using the technique shown by this recipe.

Reporting on AD users

In the previous recipe, you created a report on computer accounts that may be of interest. User and group accounts are also worth tracking. If a user has not logged on for a reasonable period, the account could be a security risk. Likewise, a user with membership in a privileged account (for example, Enterprise Admins) could be used by an attacker. IT professionals know how much easier it is just to put someone in a high-privilege group than to set up more fine-grained permissions using something like Just Enough Administration (see the *Implementing Just Enough Administration (JEA)* recipe in *Chapter 8*, *Implementing Enterprise Security*).

Regular reporting can help focus on accounts that could be deactivated, removed from a security group, or removed altogether.

In this recipe, you obtain all the accounts in the AD and examine potential security risks.

Getting ready

You run this recipe on `DC1`, a DC in the `Reskit.Org` domain, after running the recipes in this chapter. You should also have installed PowerShell 7 and VS Code on this host.

How to do it...

1. Defining a function `Get-ReskitUser` to return objects related to users in Reskit.Org domain

```
Function Get-ReskitUser {
# Get PDC Emulator DC
$PrimaryDC = Get-ADDomainController -Discover -Service PrimaryDC
# Get Users
$ADUsers = Get-ADUser -Filter * -Properties * -Server $PrimaryDC
# Iterate through them and create $Userinfo hash table:
Foreach ($ADUser in $ADUsers) {
    # Create a userinfo HT
    $UserInfo = [Ordered] @{}
    $UserInfo.SamAccountname = $ADUser.SamAccountName
    $Userinfo.DisplayName    = $ADUser.DisplayName
    $UserInfo.Office         = $ADUser.Office
    $Userinfo.Enabled        = $ADUser.Enabled
    $userinfo.LastLogonDate  = $ADUser.LastLogonDate
    $UserInfo.ProfilePath    = $ADUser.ProfilePath
    $Userinfo.ScriptPath     = $ADUser.ScriptPath
    $UserInfo.BadPWDCount    = $ADUser.badPwdCount
    New-Object -TypeName PSObject -Property $UserInfo
    }
} # end of function
```

2. Getting the users

```
$RKUsers = Get-ReskitUser
```

3. Building the report header

```
$RKReport =  ''  # first line of the report
$RkReport += "*** Reskit.Org AD Report`n"
$RKReport += "*** Generated [$(Get-Date)]`n"
$RKReport += "*******************************`n`n"
```

4. Reporting on disabled users

```
$RkReport += "*** Disabled Users`n"
$RKReport += $RKUsers |
    Where-Object {$_.Enabled -NE $true} |
        Format-Table -Property SamAccountName, Displayname |
            Out-String
```

5. Reporting on users who have not recently logged on

```
$OneWeekAgo = (Get-Date).AddDays(-7)
$RKReport += "`n*** Users Not logged in since $OneWeekAgo`n"
$RkReport += $RKUsers |
  Where-Object {$_.Enabled -and $_.LastLogonDate -le $OneWeekAgo} |
        Sort-Object -Property LastlogonDate |
            Format-Table -Property SamAccountName,lastlogondate |
                Out-String
```

6. Discovering users with a high number of invalid password attempts

   ```
   $RKReport += "`n*** High Number of Bad Password Attempts`n"
   $RKReport += $RKUsers | Where-Object BadPwdCount -ge 5 |
     Format-Table -Property SamAccountName, BadPwdCount |
       Out-String
   ```

7. Adding another report header line for this part of the report and creating an empty array of privileged users

   ```
   $RKReport += "`n*** Privileged  User Report`n"
   $PUsers = @()
   ```

8. Querying the Enterprise Admins/Domain Admins/Scheme Admins groups for members and adding them to the `$PUsers` array

   ```
   # Get Enterprise Admins group members
   $Members = Get-ADGroupMember -Identity 'Enterprise Admins' -Recursive |
       Sort-Object -Property Name
   $PUsers += foreach ($Member in $Members) {
       Get-ADUser -Identity $Member.SID -Properties * |
           Select-Object -Property Name,
                  @{Name='Group';expression={'Enterprise Admins'}},
                  whenCreated,LastLogonDate
   }
   # Get Domain Admins group members
   $Members =
     Get-ADGroupMember -Identity 'Domain Admins' -Recursive |
       Sort-Object -Property Name
   $PUsers += Foreach ($Member in $Members)
       {Get-ADUser -Identity $member.SID -Properties * |
           Select-Object -Property Name,
                   @{Name='Group';expression={'Domain Admins'}},
                   WhenCreated, LastLogonDate,SamAccountName
   }
   # Get Schema Admins members
   ```

```
$Members =
  Get-ADGroupMember -Identity 'Schema Admins' -Recursive |
    Sort-Object Name
$PUsers += Foreach ($Member in $Members) {
    Get-ADUser -Identity $member.SID -Properties * |
        Select-Object -Property Name,
            @{Name='Group';expression={'Schema Admins'}}, `
            WhenCreated, Lastlogondate,SamAccountName
}
```

9. Adding the special users to the report

```
$RKReport += $PUsers | Out-String
```

10. Displaying the final report

```
$RKReport
```

How it works...

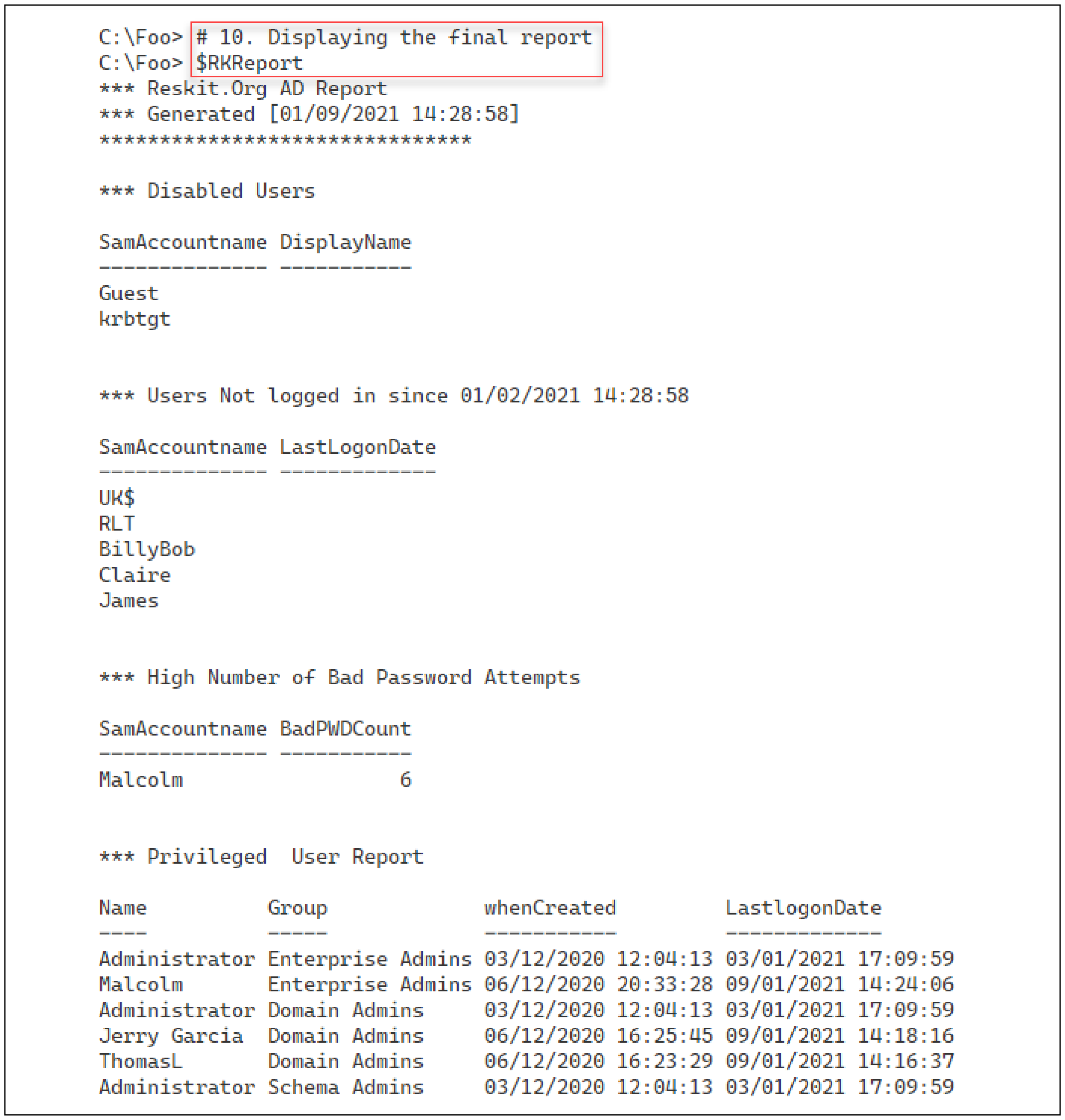
In this recipe, all the steps except the last one produce no output. The steps create a report which you view in the final step.

In *step 1*, you create a function, `Get-ReskitUser`, which creates a set of user objects related to each of the users in your AD. In *step 2*, you use the function to populate an array, `$RKUsers`, containing users and necessary details needed for your report.

With *step 3*, you build the header for the report, and then in *step 4*, you build a report section on disabled users. You use *step 4* to add details of disabled user accounts. In *step 5*, you add details of users that have not logged on within the last 7 days. In *step 6*, you add details of users who have had more than five unsuccessful login attempts.

The final section of the report lists users that are members of crucial AD groups. With *step 7*, you create a header for this section. Then with *step 8* and *step 9*, you add details of members of these groups.

Once these steps are complete, in *step 10*, you can view the output of the report, which looks like this:



```
C:\Foo> # 10. Displaying the final report
C:\Foo> $RKReport
*** Reskit.Org AD Report
*** Generated [01/09/2021 14:28:58]
*******************************

*** Disabled Users

SamAccountname DisplayName
-------------- -----------
Guest
krbtgt


*** Users Not logged in since 01/02/2021 14:28:58

SamAccountname LastLogonDate
-------------- -------------
UK$
RLT
BillyBob
Claire
James


*** High Number of Bad Password Attempts

SamAccountname BadPWDCount
-------------- -----------
Malcolm                  6


*** Privileged  User Report

Name          Group             whenCreated         LastlogonDate
----          -----             -----------         -------------
Administrator Enterprise Admins 03/12/2020 12:04:13 03/01/2021 17:09:59
Malcolm       Enterprise Admins 06/12/2020 20:33:28 09/01/2021 14:24:06
Administrator Domain Admins     03/12/2020 12:04:13 03/01/2021 17:09:59
Jerry Garcia  Domain Admins     06/12/2020 16:25:45 09/01/2021 14:18:16
ThomasL       Domain Admins     06/12/2020 16:23:29 09/01/2021 14:16:37
Administrator Schema Admins     03/12/2020 12:04:13 03/01/2021 17:09:59
```

Figure 6.49: Displaying the final report

There's more...

In *step 1*, you create a function to retrieve users. This function allows you to reformat the properties as needed, improving the output in your report.

In *step 4* through *step 8*, you build the report contents. To create the output you see in the above output, you must log in to a host in the domain. Otherwise, these sections would be blank and contain no users.

In *step 9*, you see the final report. Note that the user Malcolm both had a high number of failed login attempts and has logged in at some point successfully. If you log in using a domain account and an incorrect password, AD rejects the login and increases the bad attempt count. Once you log in successfully, however, the bad logon count is zeroed.