

EXPERT INSIGHT

Mastering Active Directory

Design, deploy, and protect Active Directory
Domain Services for Windows Server 2022



Third Edition

Dishan Francis

Packt



Preface

Microsoft Active Directory is the most widely used identity management solution. It can centrally manage identities across its infrastructure. It is equipped with different role services, features, and components that help us handle identities securely and effectively according to business requirements. For the last 20 years, Microsoft has continued improving Active Directory, and Active Directory 2022 further consolidates its approach in terms of rectifying industry requirements and protecting identity infrastructures from emerging security threats. However, a technology-rich product is not simply going to make a productive, reliable, scalable, and secure identity infrastructure. It requires knowledge of Active Directory roles services, components, and features. It also requires knowledge of how to use those effectively to match different operational requirements. Only then can we plan, design, manage, and maintain a robust identity infrastructure. Over the past few years, more and more organizations have adopted cloud technologies for a variety of reasons. With the growth of the cloud footprint, organizations' identity requirements have also changed. We can no longer limit corporate identities to on-prem infrastructures. By using Microsoft Azure Active Directory, we can extend our on-prem identities to the cloud. The hybrid AD approach provides lots of benefits for modern authentication requirements. However, security-wise, it also opens up a whole new level of challenges. Therefore, the majority of new content in the third edition is related to designing the Azure AD hybrid cloud, securing a hybrid AD environment, and protecting sensitive data.

Who this audiobook is for

If you are an Active Directory administrator, system administrator, or network professional who has basic knowledge of Active Directory and is looking to become an expert in this topic, this book is for you.

What this audiobook covers

- *Chapter 1, Active Directory Fundamentals*, explains what Active Directory is and its capabilities. This chapter also explains the main components (physical and logical structure), object types, and role services of Active Directory. Last but not least, this chapter also covers why we need an advanced identity management solution such as Azure Active Directory.
- *Chapter 2, Active Directory Domain Services 2022*, explains what we can expect with **Active Directory Domain Services (AD DS)** 2022 and how we can use the features introduced in AD DS 2016 (as there is no new **Domain Functional Level (DFL)** or **Forest Functional Level (FFL)**) to improve your existing Active Directory environment.
- *Chapter 3, Designing an Active Directory Infrastructure*, talks about what needs to be considered in Active Directory infrastructure design. This chapter discusses how to place the AD DS logical and physical components in the AD DS environment according to best practices. It also covers the design concepts for hybrid identity.

- *Chapter 4, Active Directory Domain Name System*, explains how DNS works with AD DS. This chapter also includes information about the DNS server component, different types of DNS records, zones, DNS delegation, and DNS policies.
- *Chapter 5, Placing Operations Master Roles*, talks about the **Flexible Single Master Operations (FSMO)** roles and their responsibilities. This chapter also describes things we need to consider when placing FSMO roles in an Active Directory environment.
- *Chapter 6, Migrating to Active Directory 2022*, covers the different AD DS deployment models. This chapter also provides a step-by-step guide to migrating from an older version of AD DS to AD DS 2022.
- *Chapter 7, Managing Active Directory Objects*, discusses how to create objects, find objects, modify objects, and remove objects (small-scale and large-scale) by using built-in Active Directory management tools and PowerShell commands.
- *Chapter 8, Managing Users, Groups, and Devices*, further explores the Active Directory objects by deep diving into attributes, managed service accounts, and management of different object types. Last but not least, you will also learn how to sync custom attributes to Azure Active Directory.
- *Chapter 9, Designing the OU Structure*, teaches you how to design the **organizational unit (OU)** structure properly, using different models to suit business requirements. This chapter also describes how to create, update, and remove OUs. Furthermore, this chapter also discusses how we can delegate AD administration by using OUs.
- *Chapter 10, Managing Group Policies*, mainly discusses Group Policy objects and their capabilities. Group Policy processing in an AD environment depends on many different things. In this chapter, we will deep dive into group policy processing to understand the technology behind it. We are also going to look into the different methods we can use for group policy filtering. Last but not least, we will learn about most commonly used group policies.
- *Chapter 11, Active Directory Services – Part 01*, walks us through the more advanced Active Directory topics, such as AD **Lightweight Directory Services (LDS)**, Active Directory replication, and Active Directory sites.
- *Chapter 12, Active Directory Services – Part 02*, sees you learn about Active Directory trusts in detail. This chapter also covers topics such as Active Directory database maintenance, **Read-Only Domain Controller (RODC)**, AD DS backup, and recovery.
- *Chapter 13, Active Directory Certificate Services*, discusses the planning, deployment, and maintenance of Active Directory Certificate Services. Furthermore, we will also learn how signing, encryption, and decryption work in a **public key infrastructure (PKI)**.

- *Chapter 14, Active Directory Federation Services*, focuses on **Active Directory Federation Services (AD FS)** such as planning, designing, deployment, and maintenance. This chapter also covers new features of AD FS, such as built-in Azure MFA support. At the end you will also learn how to establish a federated connection with Azure AD.
- *Chapter 15, Active Directory Rights Management Services*, covers the **Active Directory Rights Management Service (AD RMS)** role, which we can use to protect sensitive data in a business. Data is the new oil, and the value of data keeps increasing. Therefore, protection of data is important for every business. In this chapter, we will learn how AD RMS works and how to configure it.
- *Chapter 16, Active Directory Security Best Practices*, covers the protection of the Active Directory environment. Recent attacks and studies prove that adversaries are increasingly targeting identities. So, we need to be mindful of protecting our Active Directory infrastructure at any cost. In this chapter, we will learn about different tools, services, and methods we can use to protect the Active Directory environment such as Secure LDAP, Microsoft LAPS, delegated permissions, restricted RDP, and Azure AD password protection.
- *Chapter 17, Advanced AD Management with PowerShell*, is full of PowerShell scripts that can be used to manage, secure, and audit an Active Directory environment. We will also learn about the Azure Active Directory PowerShell for Graph module, which we can use to manage, query, and update AD objects in a hybrid AD environment.
- *Chapter 18, Hybrid Identity*, discusses how we can extend our on-prem AD DS infrastructure to Azure Active Directory. Before we work on the implementation, we will deep dive into the planning process of the Azure AD hybrid setup. In this chapter, we will also learn about different Azure AD connects deployment models, Azure AD cloud sync, Secure LDAP, and replica sets.
- *Chapter 19, Active Directory Audit and Monitoring*, teaches you how to monitor your on-prem/hybrid AD DS infrastructure using different tools and methods (cloud based and on-prem). This chapter also demonstrates how to audit the health of an Active Directory environment.
- *Chapter 20, Active Directory Troubleshooting*, discusses how to troubleshoot the most common Active Directory infrastructure issues using different tools and methods. Furthermore, we will also look into the most common Azure AD Connect errors, which can have a direct impact on the health of the Azure AD hybrid environment. You can find this chapter available online at: static.packt-cdn.com/downloads/9781801070393_Chapter_20.pdf
- *Appendix A, References*, covers the *Further reading* section chapter wise. It's freely available online for our readers and here is the link: static.packt-cdn.com/downloads/Mastering_Active_Directory_References.pdf.

To get the most out of this audiobook

This book is ideal for IT professionals, system engineers, and administrators who have a basic knowledge of Active Directory Domain Services. A basic knowledge of PowerShell is also required, since most of the role deployment, configuration, and management is done by using PowerShell commands and scripts.

Download the example code files

The code bundle for the book is hosted on GitHub at github.com/PacktPublishing/Mastering-Active-Directory-Third-Edition. We also have other code bundles from our rich catalog of books and videos available at github.com/PacktPublishing/. Check them out!

Download the supplementary materials

Packt audiobooks have been selected for a seamless audio experience. Some topics, however, do come with elements like images that aren't natural for this medium. We've adapted the content of the audiobooks so that you can listen to the audio without needing to refer to these visual elements unless necessary. To give you the choice between listening to just the audio and listening to the audio while referring to the visual elements, we've created a PDF that contains all the elements that cannot translate to the audio. All references to images in the audiobook can be found within this PDF. You can download the PDF from <https://github.com/PacktPublishing/Mastering-Active-Directory-Third-Edition-Audiobook>.

Chapter 1

Links

- <https://bit.ly/3CSjC08>
- <https://bit.ly/2ZSnTIK>
- <https://bit.ly/3mNckFB>
- <https://bit.ly/3whpgGV>
- <https://bit.ly/3mMxjbu>
- <https://bit.ly/3BNw0qS>
- <https://bit.ly/3nZBpIQ>
- <https://bit.ly/3o3uqlL>
- <https://vz.to/3CQvPCl>
- <https://bit.ly/3q6wSed>
- <https://bit.ly/3ER8Isq>
- <https://ibm.co/3wwOSif>
- <https://bit.ly/3HUQWXc>
- [\(https://bit.ly/3mLiJkg](https://bit.ly/3mLiJkg)
- More information about the Nobelium attack is available in the following articles, which are published by Microsoft:
 - <https://bit.ly/3wl8fvx>
 - <https://bit.ly/3BJfbDv>
 - <https://bit.ly/3bI09Dx>
- More information about WebAuthn is available at the following links:
 - <https://bit.ly/3wkLW93>
 - <https://bit.ly/3bQfamE>
- Step-by-step guide: Azure AD password-less sign-in using FIDO2 security keys:
<https://bit.ly/3GTiHmG>
- Step-by-step guide: Enable Windows 10 password-less authentication with FIDO2 security keys (Azure AD + Microsoft Intune): <https://bit.ly/3wl8wyZ>.

- <https://mck.co/3bJK14p>
- <https://bit.ly/3ENOtvz>
- <https://bit.ly/3BQImER>
- <https://bit.ly/3BQb5cK>
- <https://bit.ly/3nYq7z3>
- <https://bit.ly/3BUNAj6>

Figures

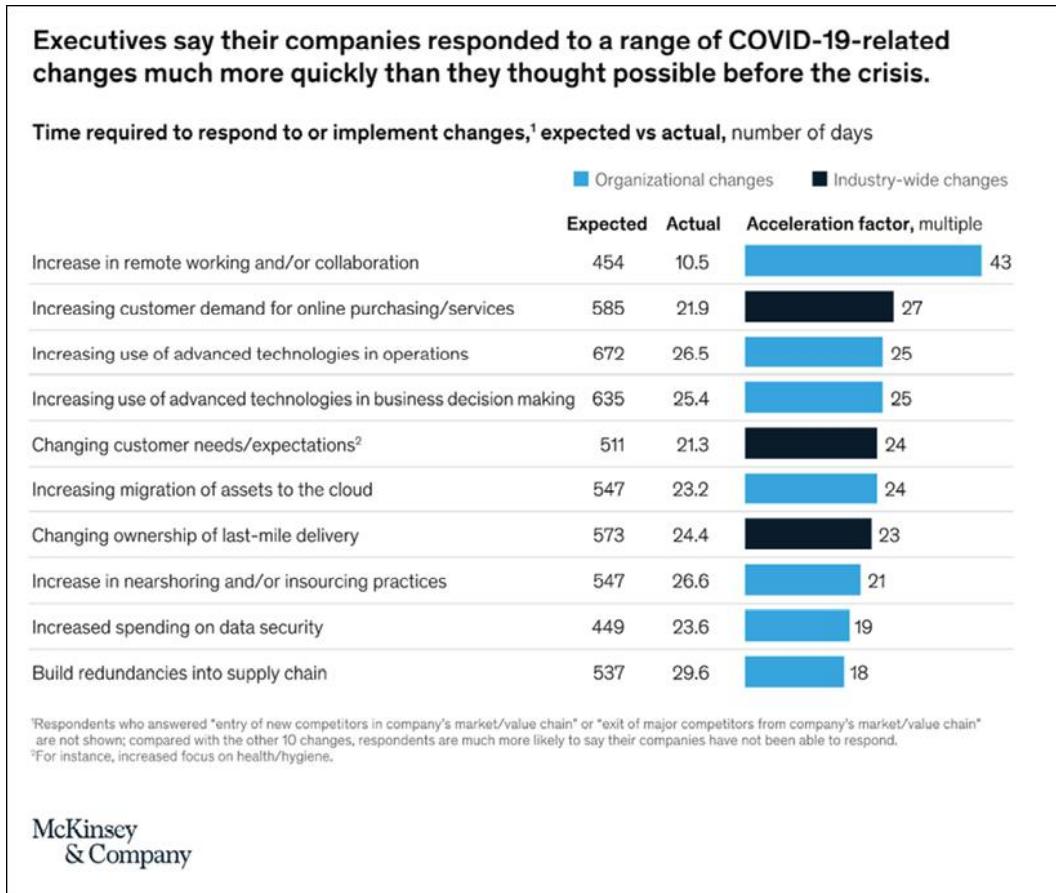


Figure 1.1: Speed of responses to pandemic challenges. Source: <https://mck.co/2Ykj9Fd>

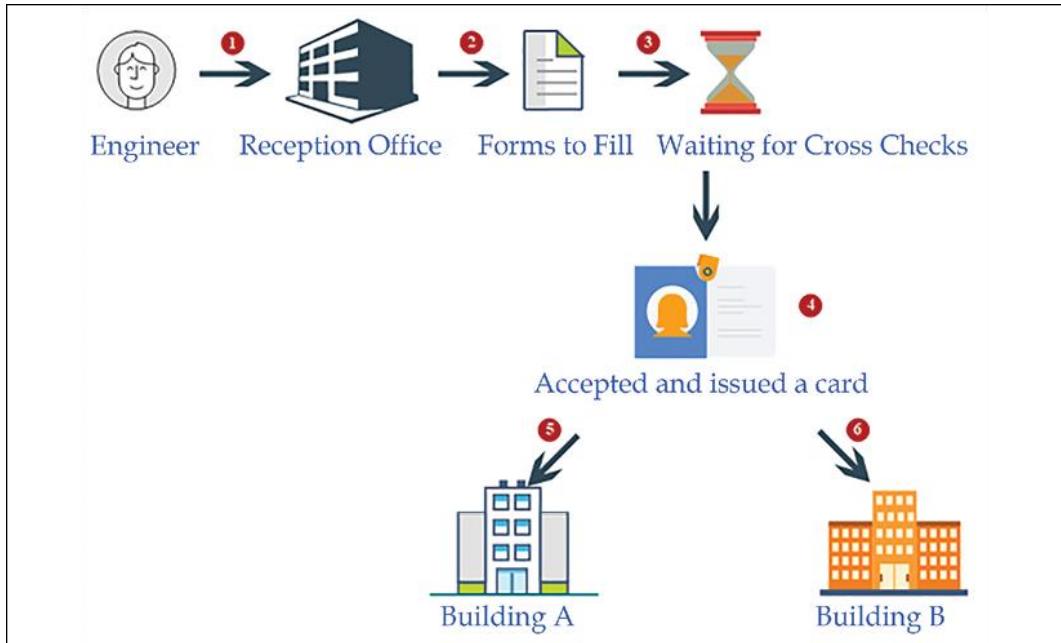


Figure 1.2: Process of entering physical headquarters

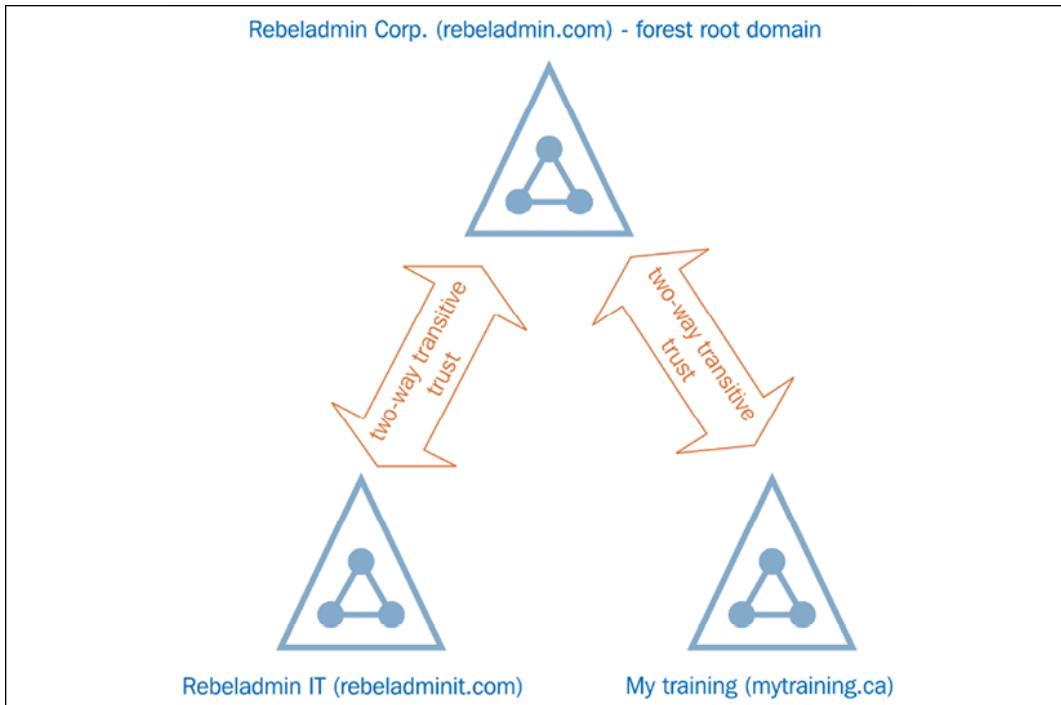


Figure 1.3: Domains in the rebeladmin.com forest

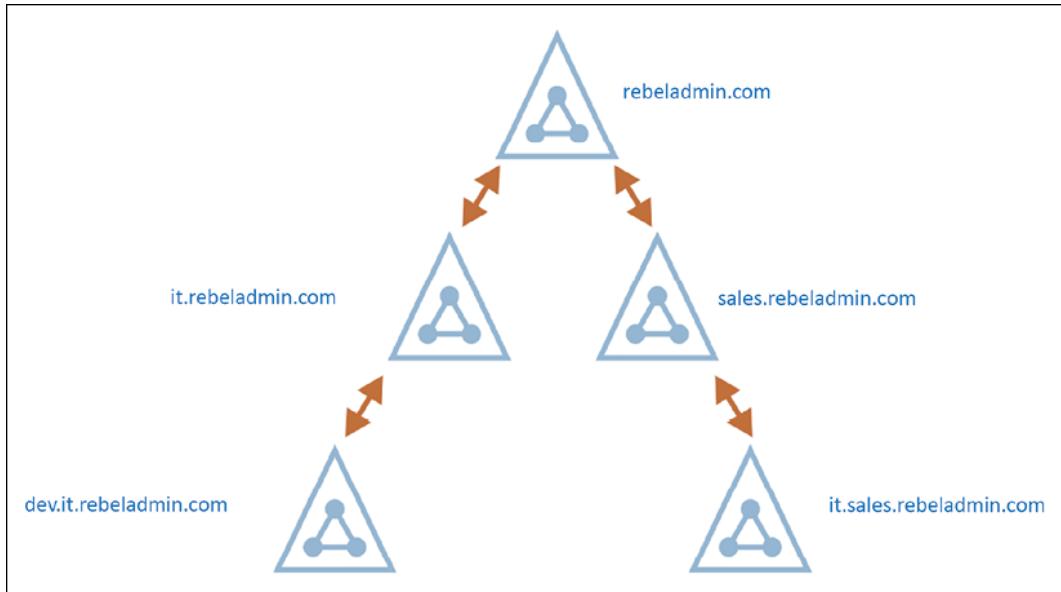


Figure 1.4: Subdomains in a contiguous namespace

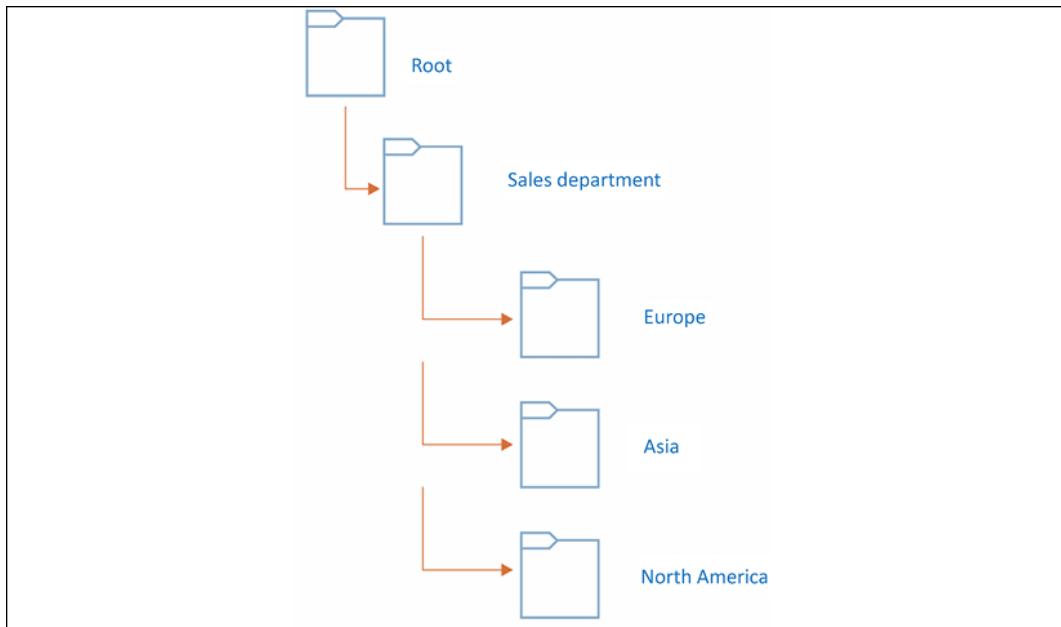


Figure 1.5: Organizational unit hierarchy

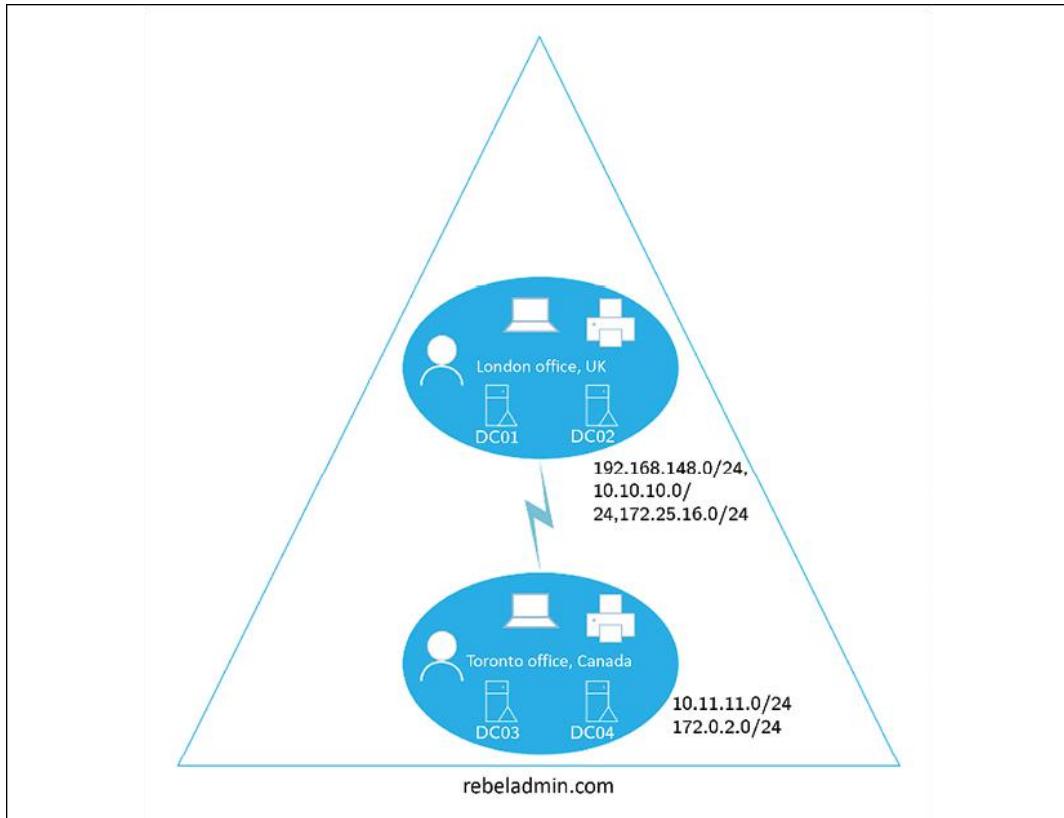


Figure 1.6: Active Directory connection

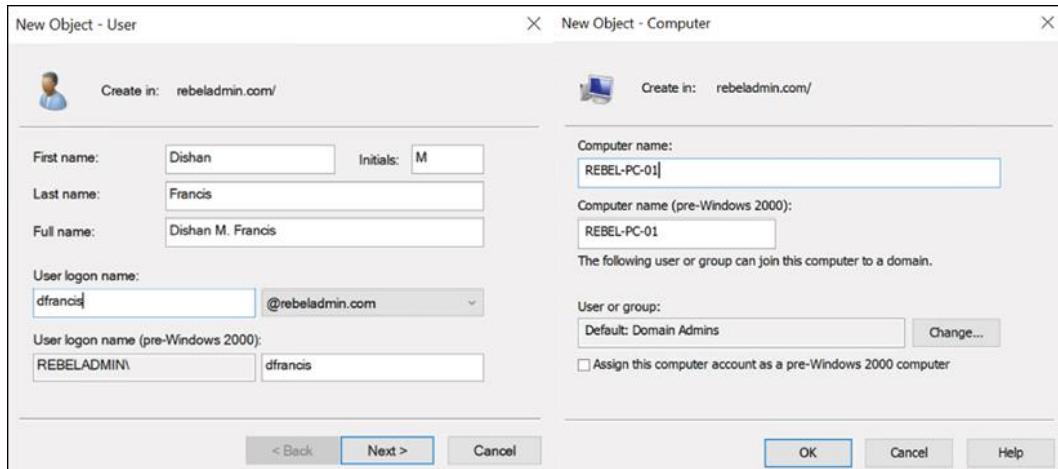


Figure 1.7: Creating new objects

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
PS C:\Users\Administrator> Get-ADUser dfrancis

DistinguishedName : CN=Dishan Francis,CN=Users,DC=rebeladmin,DC=com
Enabled : True
GivenName : Dishan
Name : Dishan Francis
ObjectClass : user
ObjectGUID : 94017e0b-d53b-4730-abf3-4c41e90de420
SamAccountName : dfrancis
SID : S-1-5-21-4041220333-1835452706-552999228-1106
Surname : Francis
UserPrincipalName : dfrancis@rebeladmin.com

PS C:\Users\Administrator>

```

Figure 1.8: ObjectGUID and sid as they appear in PowerShell

Tables

Functional Level	Domain Controller Operating System
Windows Server 2016	Windows Server 2022 Windows Server 2019 Windows Server 2016
Windows Server 2012R2	Windows Server 2022 Windows Server 2019 Windows Server 2016 Windows Server 2012 R2
Windows Server 2012	Windows Server 2022 Windows Server 2019 Windows Server 2016 Windows Server 2012 R2 Windows Server 2012
Windows Server 2008R2	Windows Server 2022 Windows Server 2019 Windows Server 2016 Windows Server 2012 R2 Windows Server 2012 Windows Server 2008 R2
Windows Server 2008	Windows Server 2022 Windows Server 2019 Windows Server 2016 Windows Server 2012 R2 Windows Server 2012 Windows Server 2008 R2 Windows Server 2008

Table 1.1: Domain Controller operating systems for each functional level

PowerShell cmdlets	Description
Install-WindowsFeature AD-Domain-Services	This cmdlet will install the AD DS role. Please note this will only install the AD DS role on the server. This is further explained in <i>Chapter 6, Migrating to Active Directory 2022</i>
Install-WindowsFeature AD FS-Federation	This cmdlet will install the AD FS role.
Install-WindowsFeature ADLDS	This cmdlet will install AD LDS.
Install-WindowsFeature ADRMS	This cmdlet will install AD RMS. This role has two subfeatures, which are AD Rights Management Server and Identity Federation Support. If required, these individual roles can be installed using Install-WindowsFeature ADRMS, ADRMS-Server, ADRMS-Identity, or Install-WindowsFeature ADRMS -IncludeAllSubFeature. It will install all the subfeatures.
Install-WindowsFeature AD-Certificate	This cmdlet will install AD CS. This role has six subroles, which are certification authority (ADCS-Cert-Authority), Certificate Enrollment Policy Web Service (ADCS-Enroll-Web-Pol), Certificate Enrollment Web Service (ADCS-Enroll-Web-Svc), Certification Authority Web Enrollment (ADCS-Web-Enrollment), Network Device Enrollment Service (ADCS-Device-Enrollment), and Online Responder (ADCS-Online-Cert). These subfeatures can be added individually or together.

Table 1.2: PowerShell cmdlets to install Active Directory server roles

Chapter 2

Links

- <https://bit.ly/3q4Re7E>
- <https://bit.ly/3BOPJMX>
- <https://bit.ly/3whrR3D>
- <https://bit.ly/3bMk6sQ>
- <https://bit.ly/3whR6Tc>

Figures

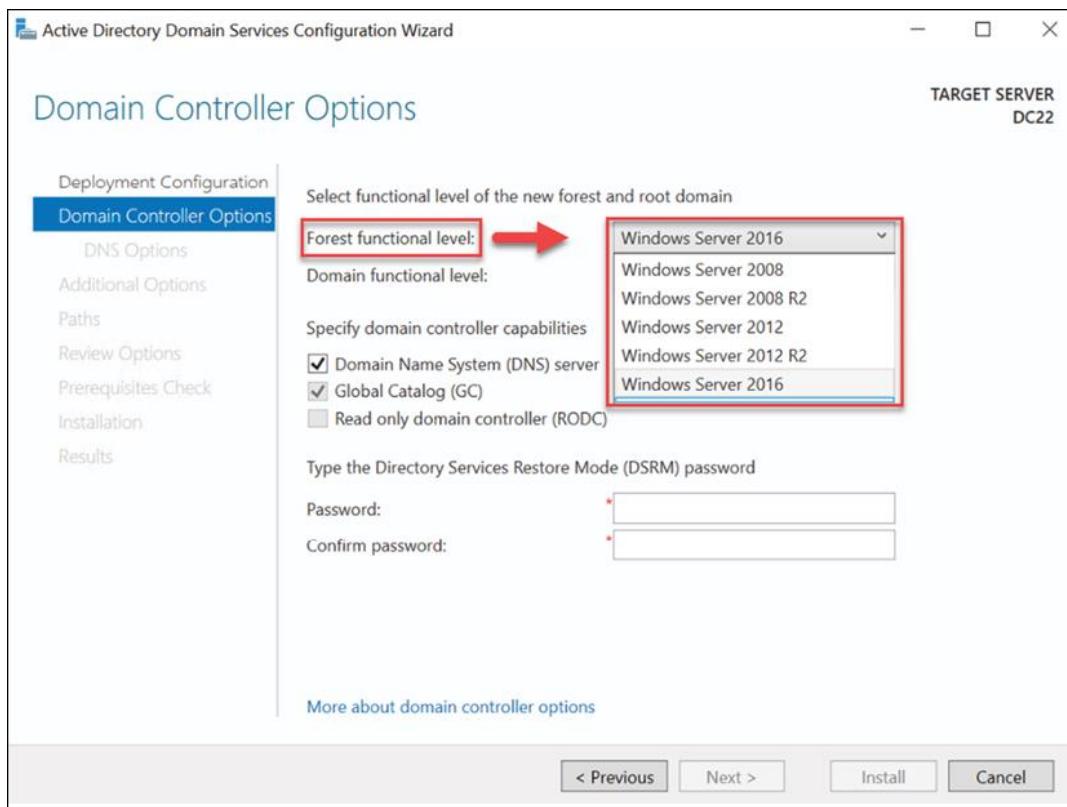


Figure 2.1: Forest functional levels for Windows Server 2019

Web attacks blocked per month

The peak level of malicious activity took place in October, when more than 27.7 million web attacks were blocked.

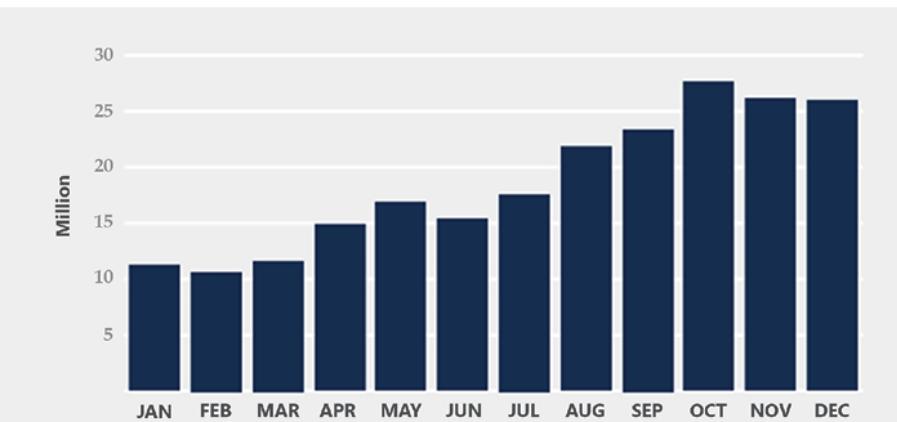


Figure 2.2: Web attacks blocked per month

3.1. Attack complexity

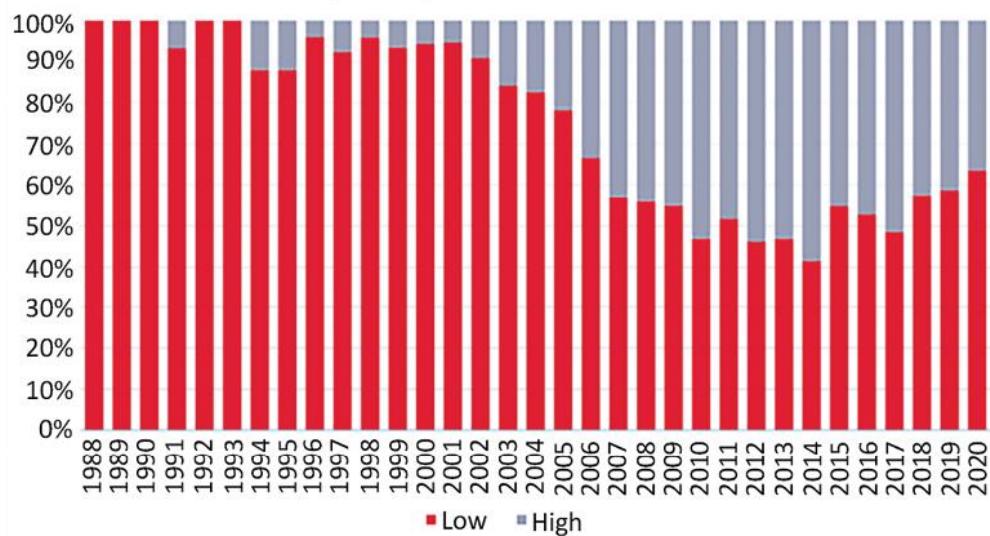


Figure 3: Percentage of low and high complexity CVEs by year: 1988-2020

Figure 2.3: Attack complexity

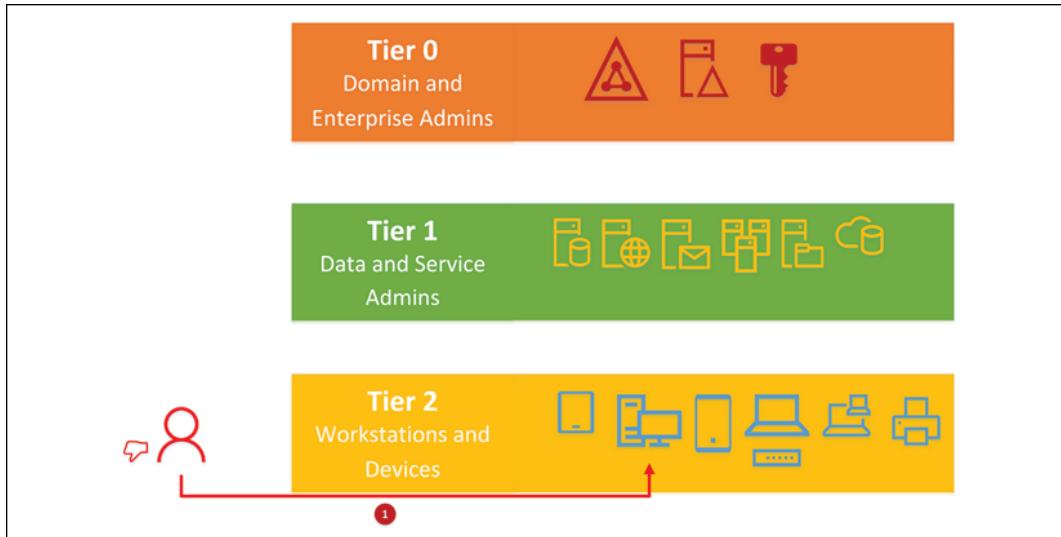


Figure 2.4: AD attack – initial breach

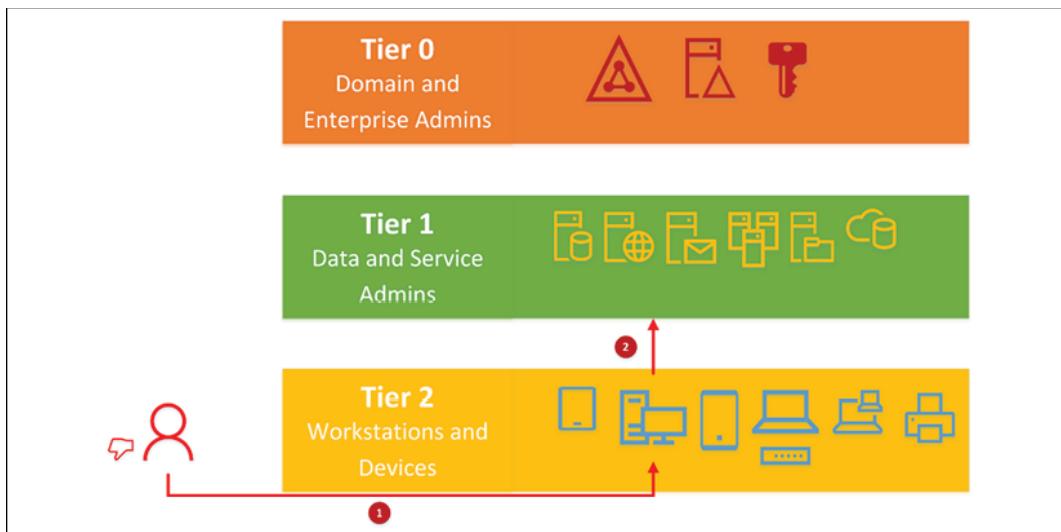


Figure 2.5: AD attack – lateral movement

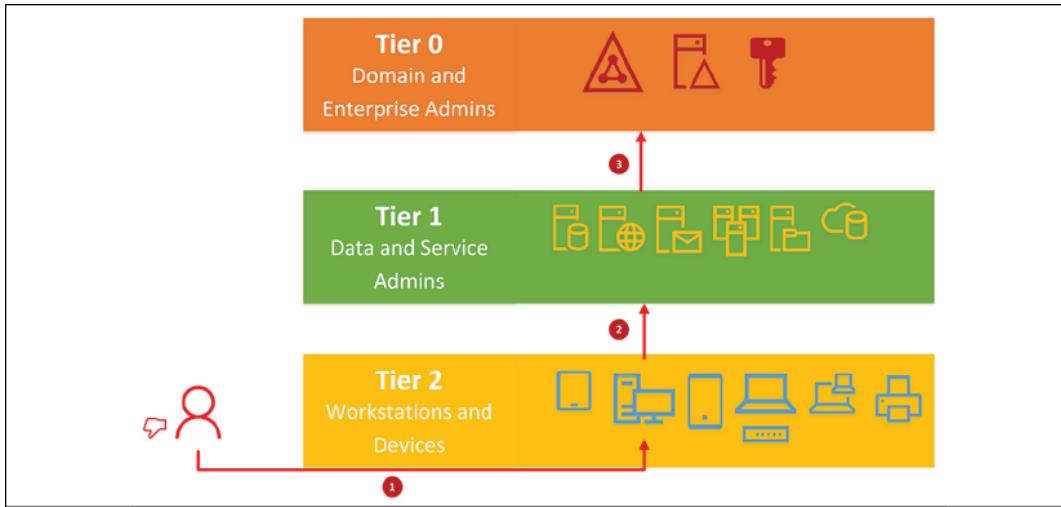


Figure 2.6: AD Attack – access to privileged accounts

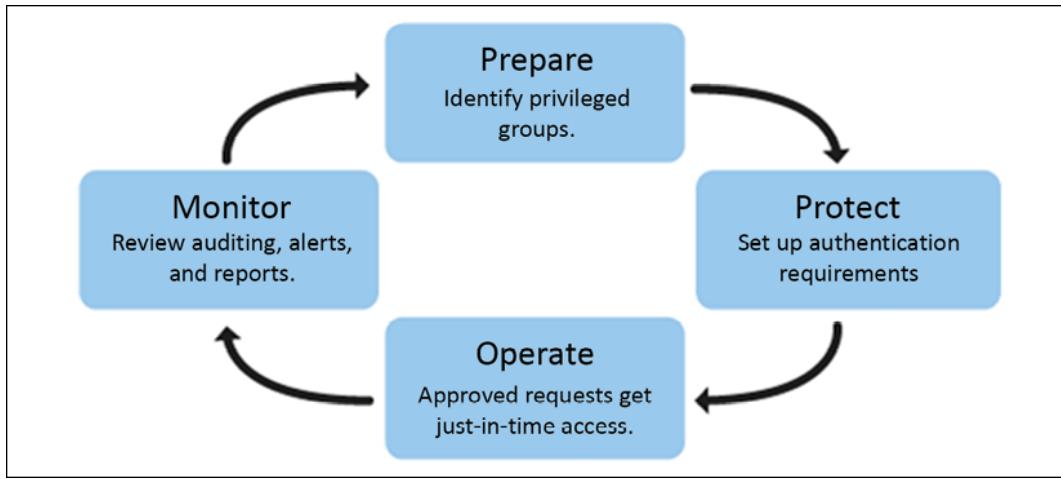


Figure 2.7: The PAM lifecycle

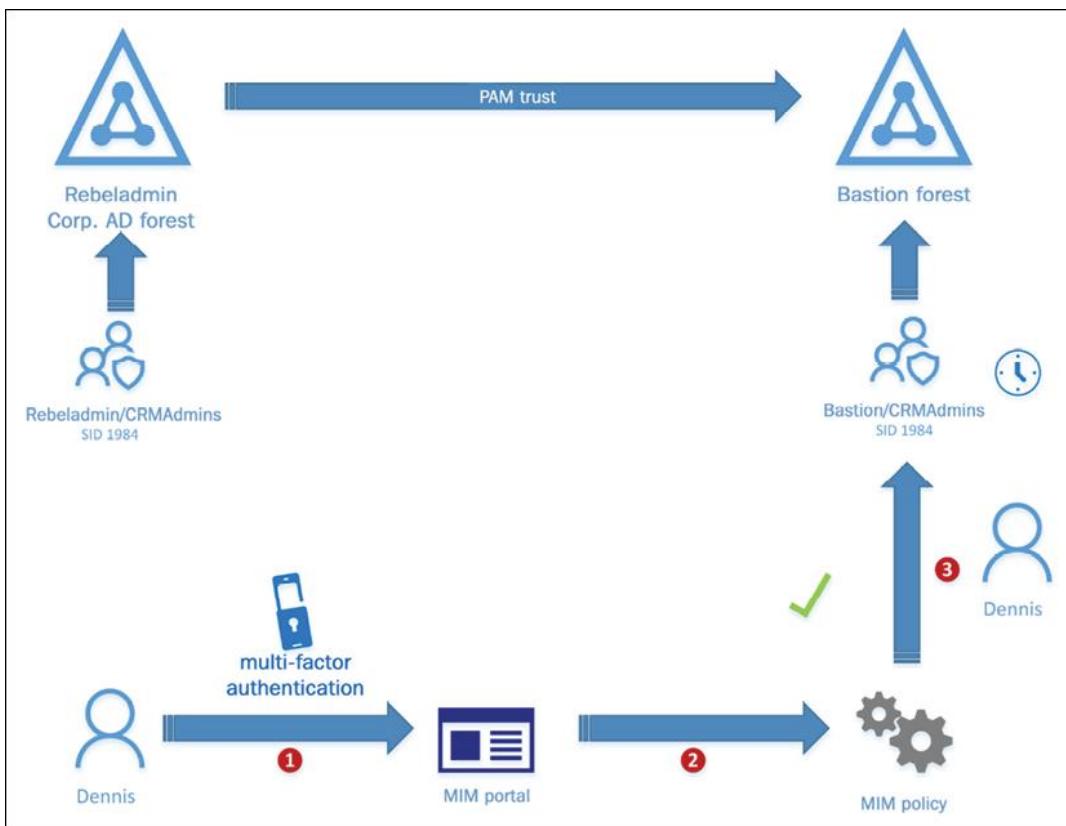


Figure 2.8: PAM in action

```
PS C:\Windows\System32> Get-ADGroupMember "Domain Admins"

distinguishedName : CN=dfrancis,CN=Users,DC=rebeladmin,DC=com
name              : dfrancis
objectClass       : user
objectGUID        : 196f45a5-d966-4063-b947-b598cd15e305
SamAccountName   : dfrancis
SID               : S-1-5-21-2368539100-1613354624-3645015003-500
```

Figure 2.9: Domain admin group membership

```
PS C:\Windows\System32> Get-ADGroup 'Domain Admins' -Property member -ShowMemberTimeToLive

DistinguishedName : CN=Domain Admins,CN=Users,DC=rebeladmin,DC=com
GroupCategory     : Security
GroupScope        : Global
member            : {<TTL=3435>,CN=Adam Curtiss,CN=Users,DC=rebeladmin,DC=com,
                   CN=dfrancis,CN=Users,DC=rebeladmin,DC=com}
Name              : Domain Admins
ObjectClass       : group
ObjectGUID        : a39fe22a-9303-41e2-9560-a7889083ca1a
SamAccountName   : Domain Admins
SID               : S-1-5-21-2368539100-1613354624-3645015003-512
```

Figure 2.10: TTL for group membership

```
Administrator: PowerShell 7 (x64)
PowerShell 7.1.2
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Users\acurtiss> klist

Current LogonId is 0:0x2168f2

Cached Tickets: (1)

#0> Client: acurtiss @ REBELADMIN.COM
Server: krbtgt/REBELADMIN.COM @ REBELADMIN.COM
Kerberos Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent name_canonicalize
Start Time: 2/28/2021 22:09:50 (local)
End Time: 2/28/2021 23:01:25 (local)
Renew Time: 2/28/2021 23:01:25 (local)
Session Key Type: AES-256-CTS-HMAC-SHA1-96
Cache Flags: 0x1 -> PRIMARY
Kdc Called: DC01
PS C:\Users\acurtiss> ■
```

Figure 2.11: Kerberos ticket renewal time

Chapter 3

Figures

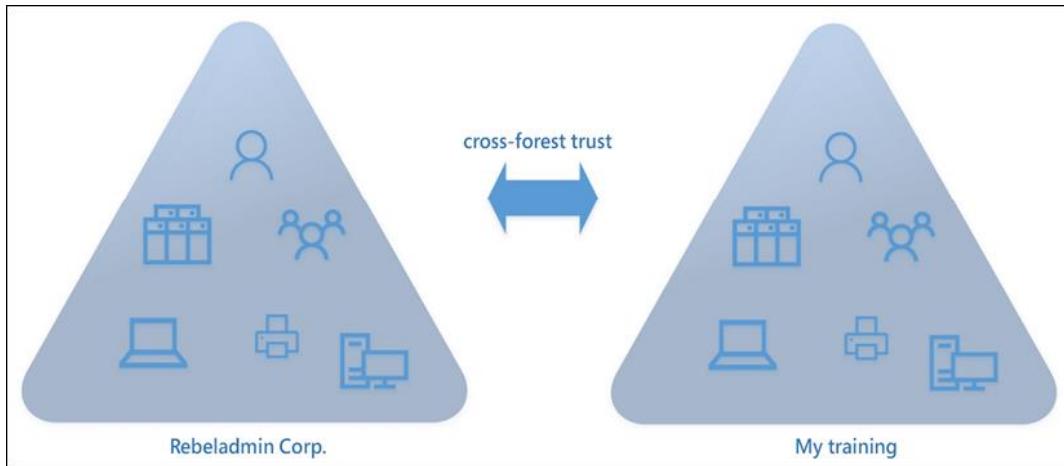


Figure 3.1: Organizational forest model example

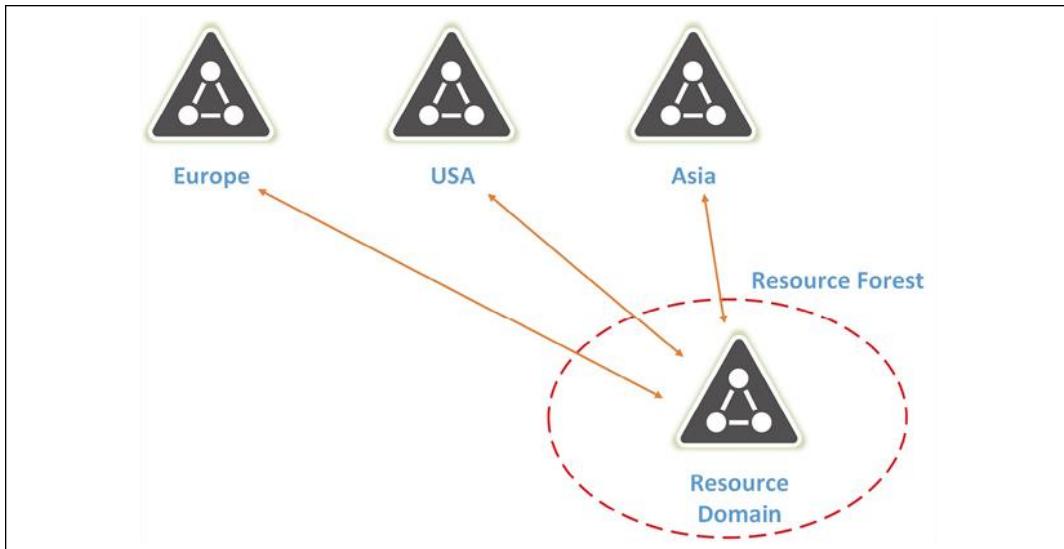


Figure 3.2: Resource forest model example 1

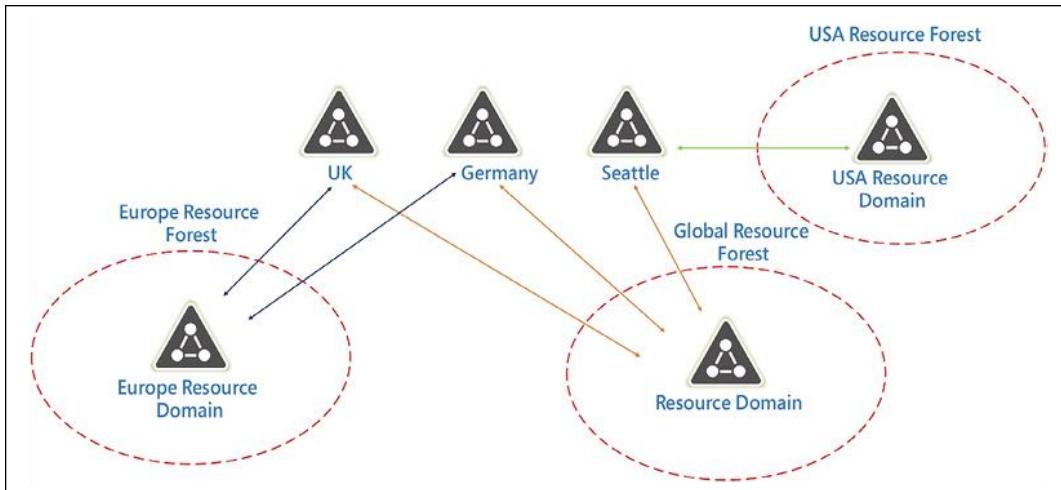


Figure 3.3: Resource forest model example 2

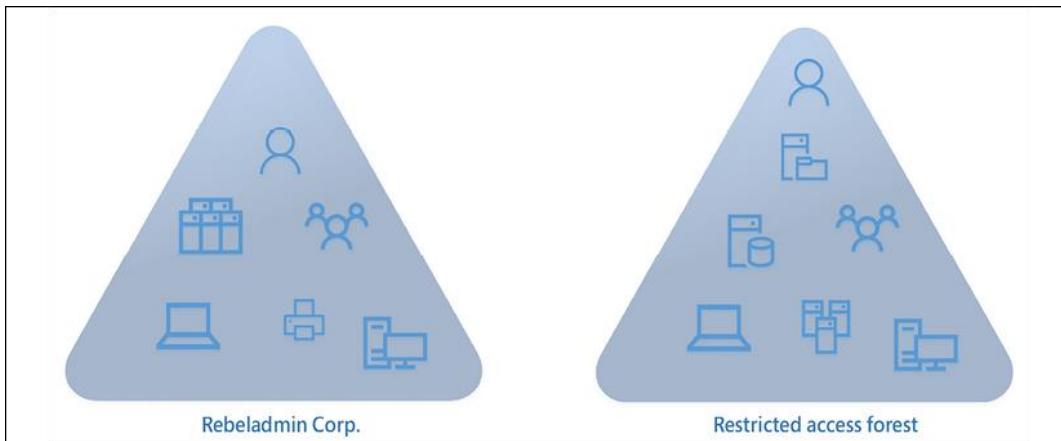


Figure 3.4: Restricted access forest model example

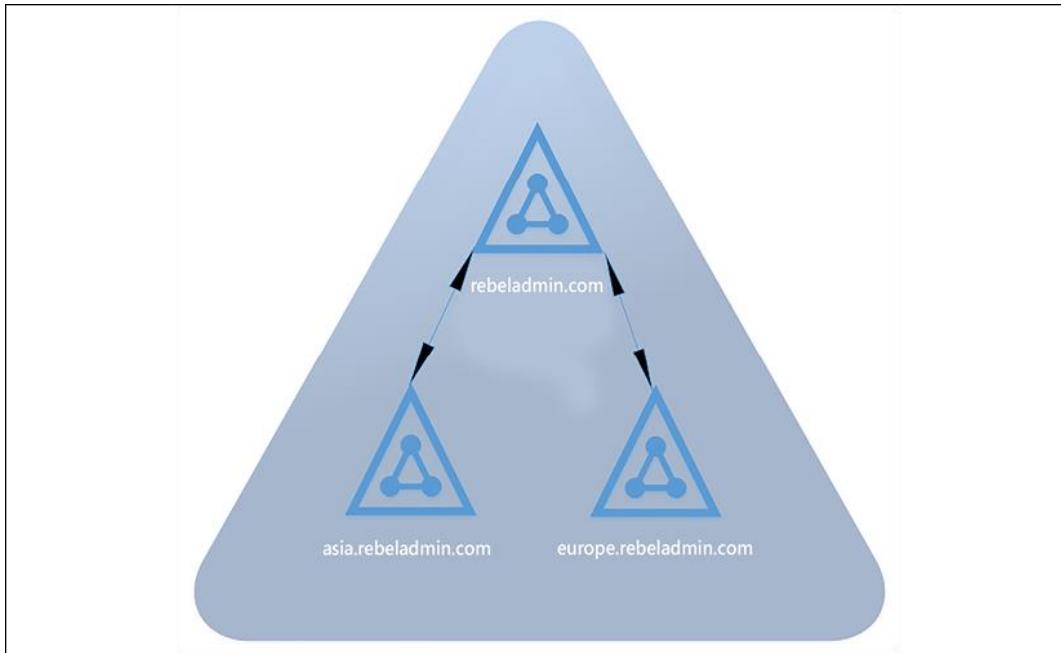


Figure 3.5: Regional domain model example

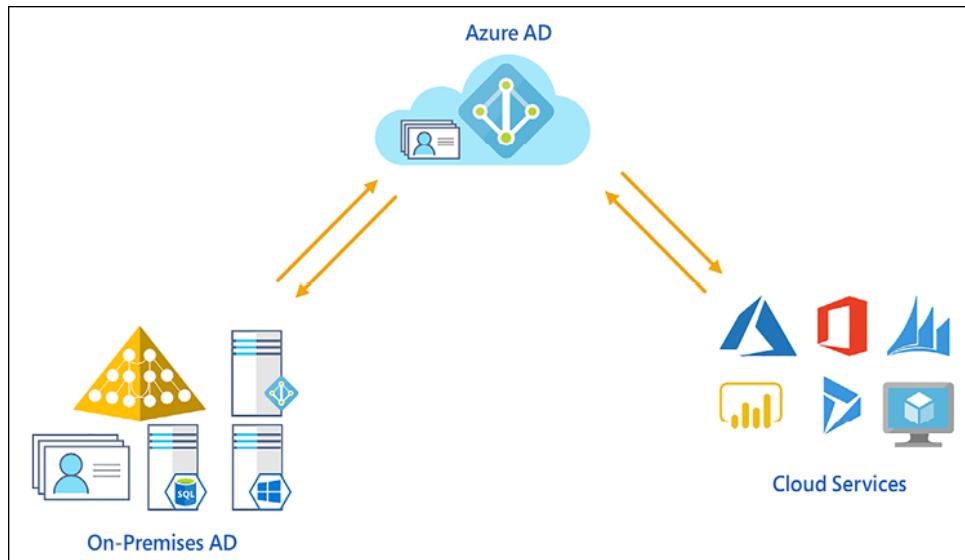
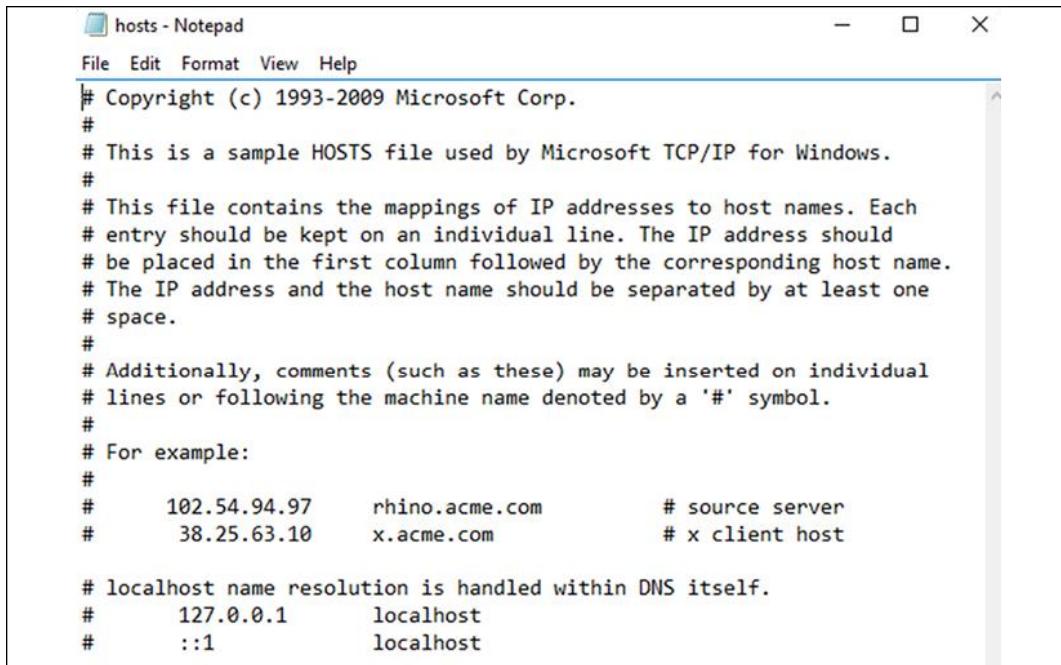


Figure 3.6: Unified access experience

Chapter 4 Figures



```
hosts - Notepad
File Edit Format View Help
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97      rhino.acme.com      # source server
#      38.25.63.10       x.acme.com        # x client host
#
# localhost name resolution is handled within DNS itself.
#      127.0.0.1        localhost
#      ::1              localhost
```

Figure 4.1: The hosts file

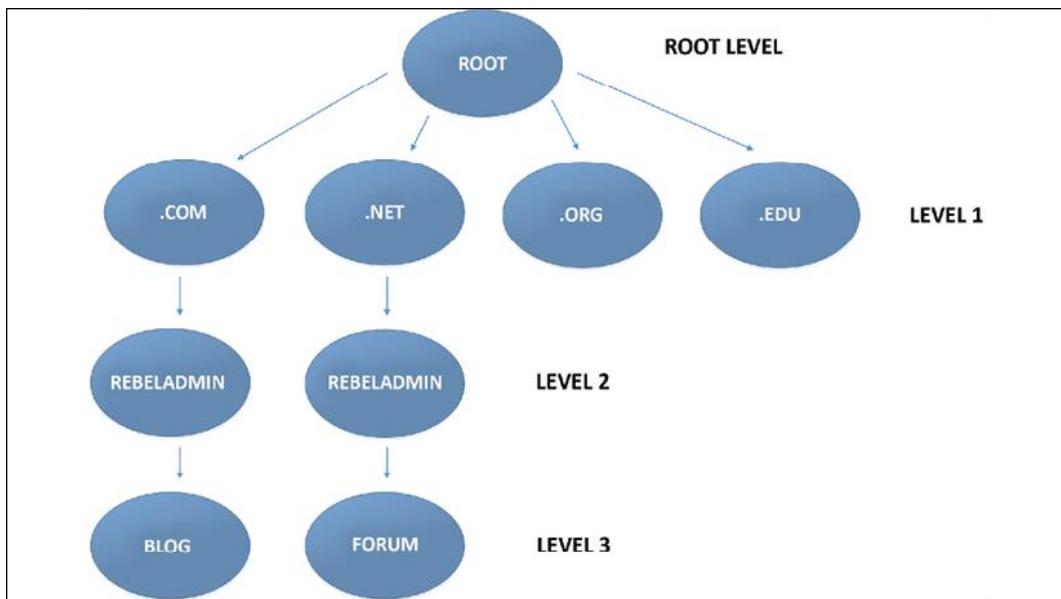


Figure 4.2: Domain naming hierarchy



Figure 4.3: Domain name example

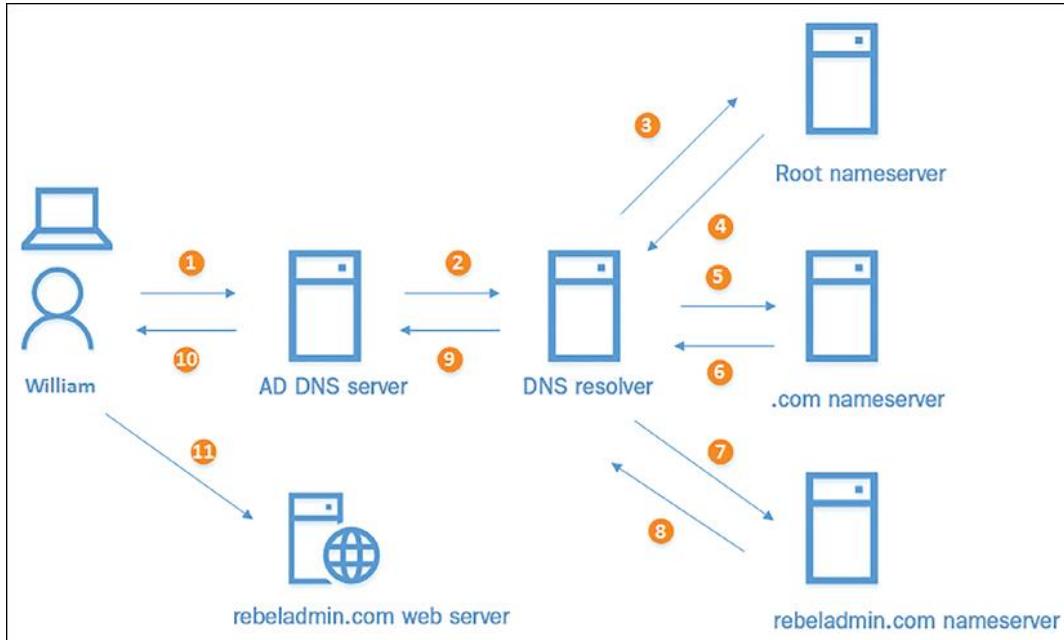


Figure 4.4: DNS query example

```
PS C:\Windows\System32> Get-DnsServerForwarder

UseRootHint      : True
Timeout(s)       : 3
EnableReordering : True
IPAddress        : 168.63.129.16
ReorderedIPAddress : 168.63.129.16

PS C:\Windows\System32>
```

Figure 4.5: DNS forwarders

```
PS C:\Windows\System32> Get-DnsServerRootHint

NameServer          IPAddress
-----
j.root-servers.net. 2001:503:c27::2:30
b.root-servers.net. 2001:500:200::b
f.root-servers.net. 2001:500:2f::f
m.root-servers.net. 2001:dc3::35
a.root-servers.net. 2001:503:ba3e::2:30
h.root-servers.net. 2001:500:1::53
d.root-servers.net. 2001:500:2d::d
c.root-servers.net. 2001:500:2::c
k.root-servers.net. 2001:7fd::1
e.root-servers.net. 2001:500:a8::e
g.root-servers.net. 2001:500:12::d0d
i.root-servers.net. 2001:7fe::53
l.root-servers.net. 2001:500:9f::42
```

Figure 4.6: Root DNS servers

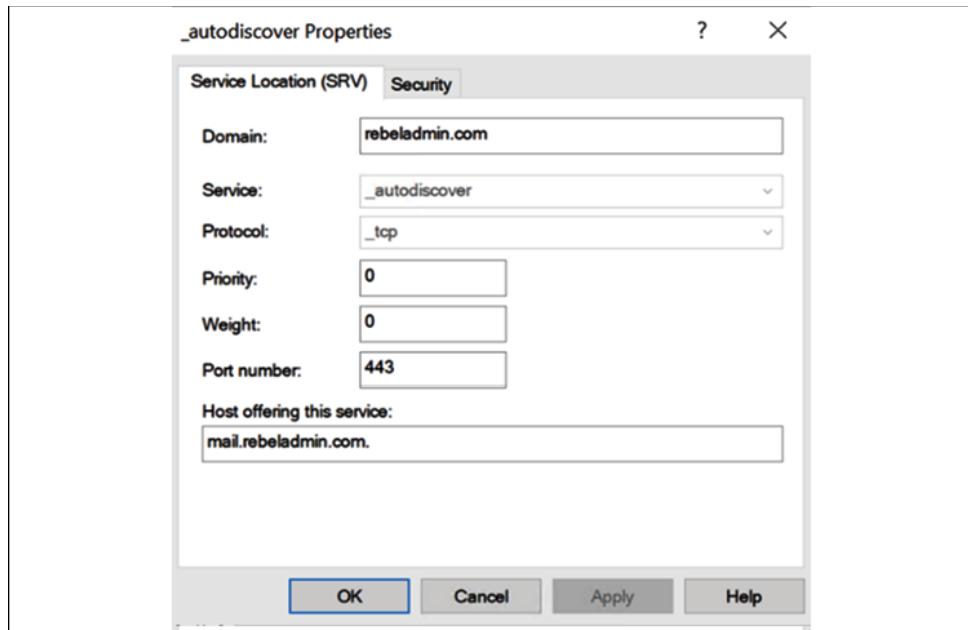


Figure 4.7: Sample SRV record

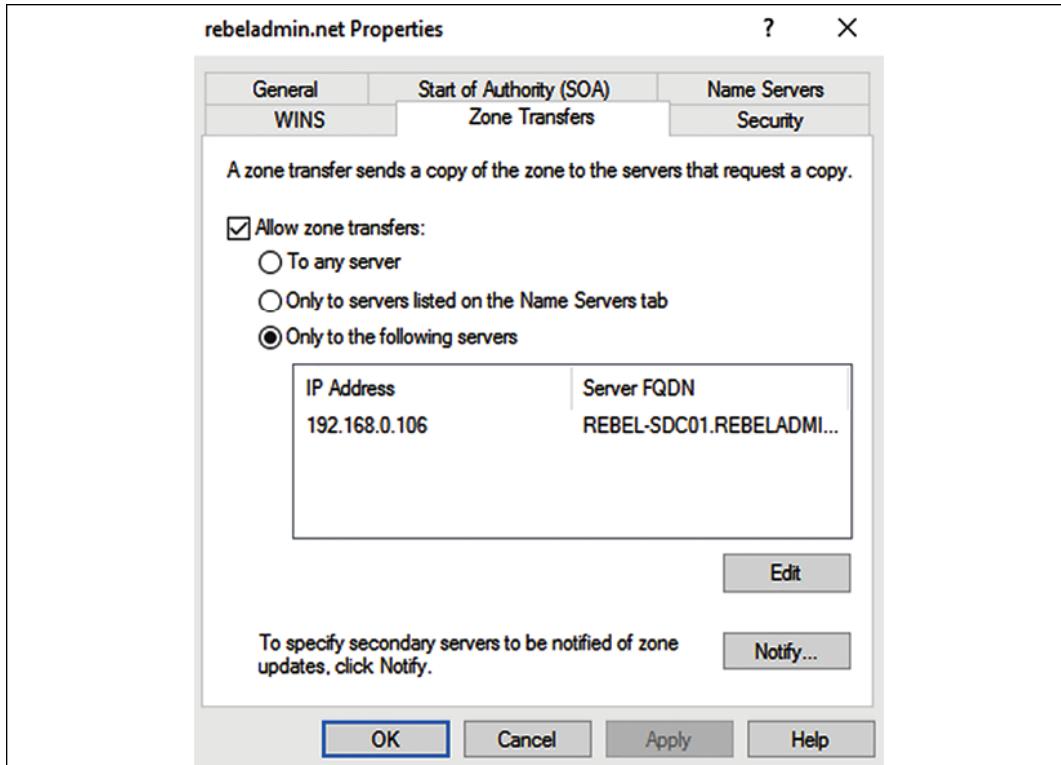


Figure 4.8: Zone transfer settings

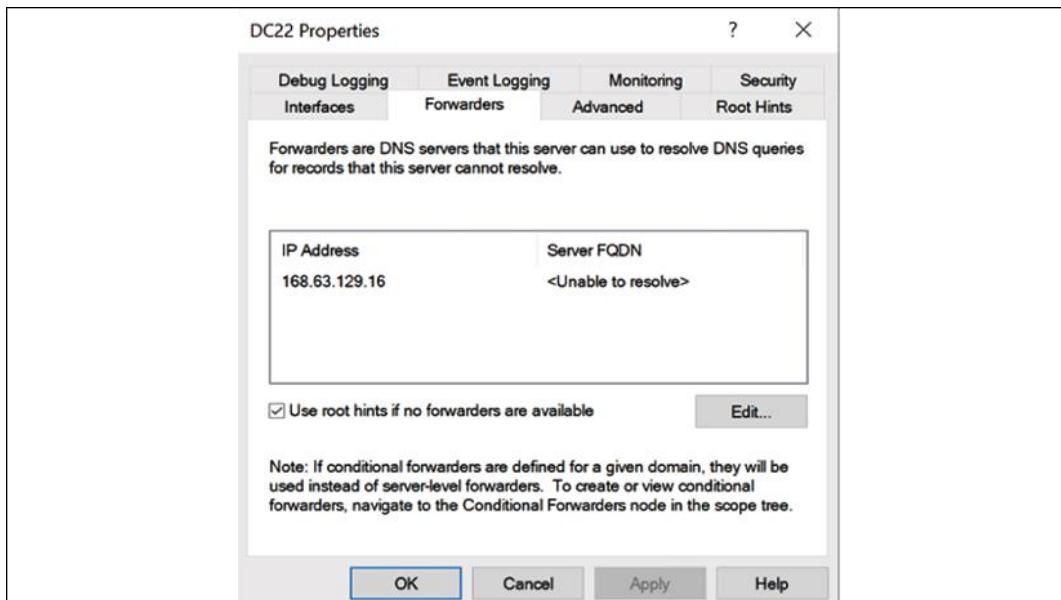
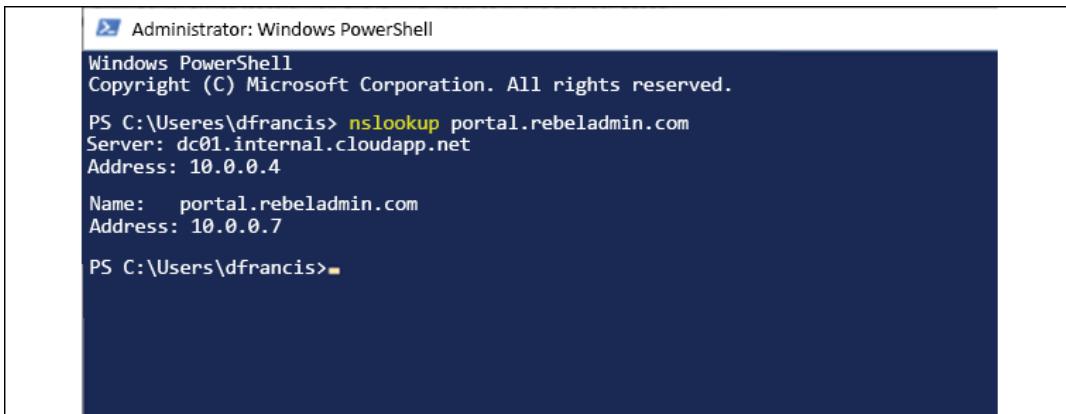


Figure 4.9: DNS forwarders



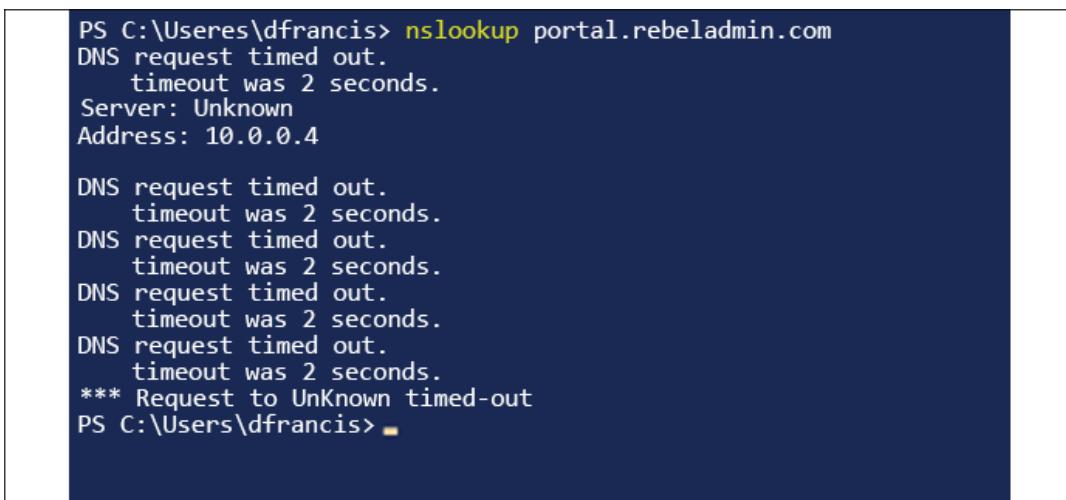
```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\dfrancis> nslookup portal.rebeladmin.com
Server: dc01.internal.cloudapp.net
Address: 10.0.0.4

Name:  portal.rebeladmin.com
Address: 10.0.0.7

PS C:\Users\dfrancis>
```

Figure 4.10: DNS query



```
PS C:\Users\dfrancis> nslookup portal.rebeladmin.com
DNS request timed out.
    timeout was 2 seconds.
Server: Unknown
Address: 10.0.0.4

DNS request timed out.
    timeout was 2 seconds.
*** Request to Unknown timed-out
PS C:\Users\dfrancis>
```

Figure 4.11: Failed DNS query

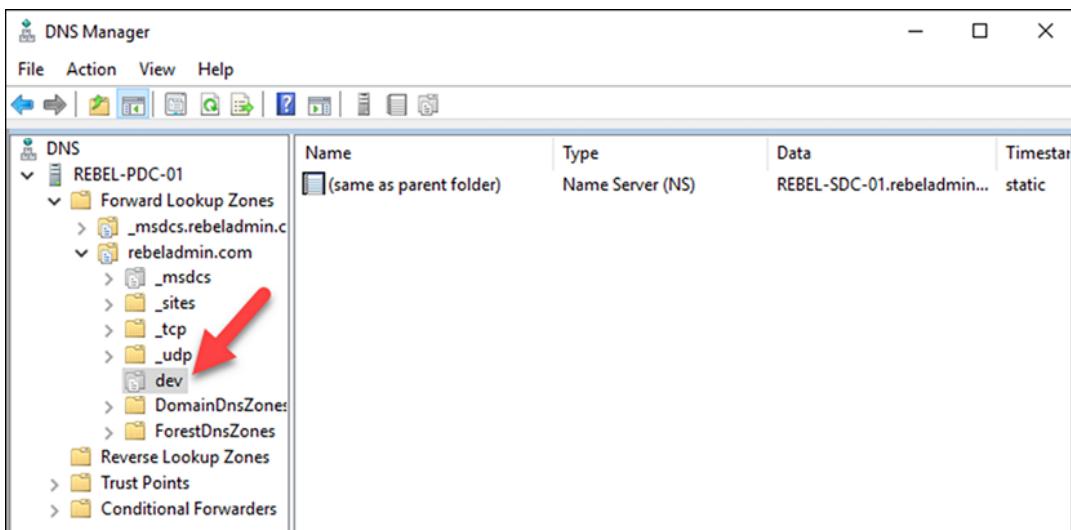


Figure 4.12: Delegated zone

The screenshot shows an 'Administrator: Windows PowerShell' window. The command 'ping app1.dev.rebeladmin.com -t' is run, and the output shows multiple replies from the IP address 192.168.0.110, indicating successful resolution of the name from the delegated zone.

```
Administrator: Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> ping app1.dev.rebeladmin.com -t

Pinging app1.dev.rebeladmin.com [192.168.0.110] with 32 bytes of data:
Reply from 192.168.0.110: bytes=32 time<1ms TTL=128
Reply from 192.168.0.110: bytes=32 time=4ms TTL=128
Reply from 192.168.0.110: bytes=32 time<1ms TTL=128
Reply from 192.168.0.110: bytes=32 time=1ms TTL=128
Reply from 192.168.0.110: bytes=32 time<1ms TTL=128
Reply from 192.168.0.110: bytes=32 time=1ms TTL=128
```

Figure 4.13: Resolve names from the delegated zone

Chapter 5

Figures



Figure 5.1: Time sources

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17763.1879]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\dfrancis>w32tm.exe /config /syncfromflags:manual /manualpeerlist:132.163.97.1,0x8 /reliable:yes /update
The command completed successfully.

C:\Users\dfrancis>w32tm.exe /config /update
The command completed successfully.

C:\Users\dfrancis>
```

Figure 5.2: Configuring the external time source

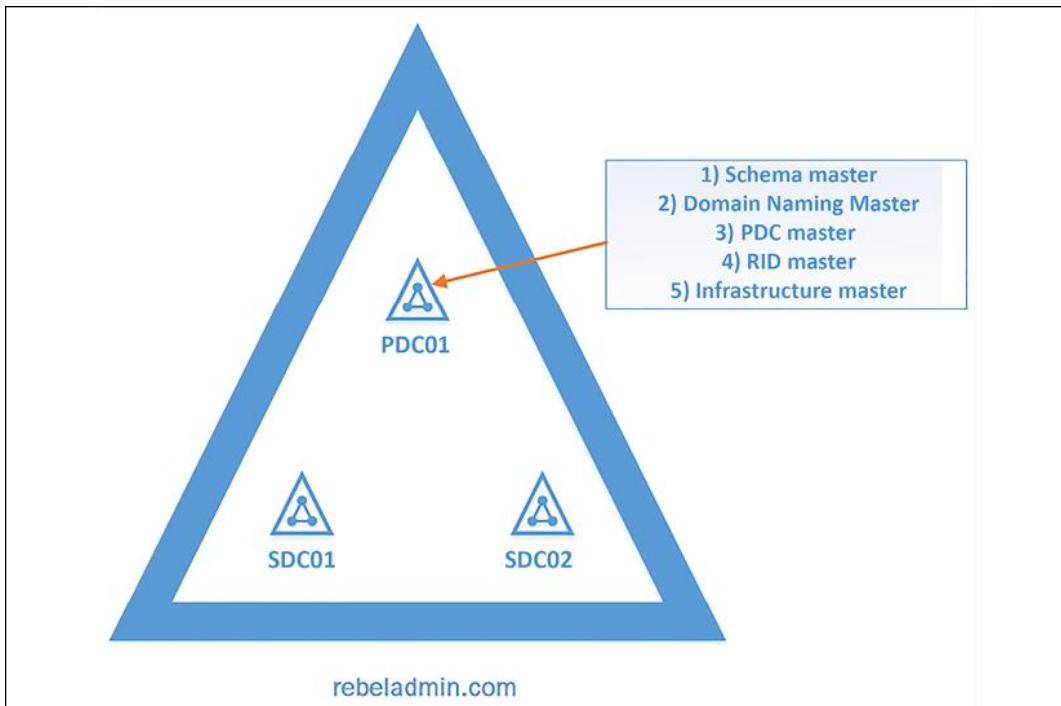


Figure 5.3: FSMO role placement example 1

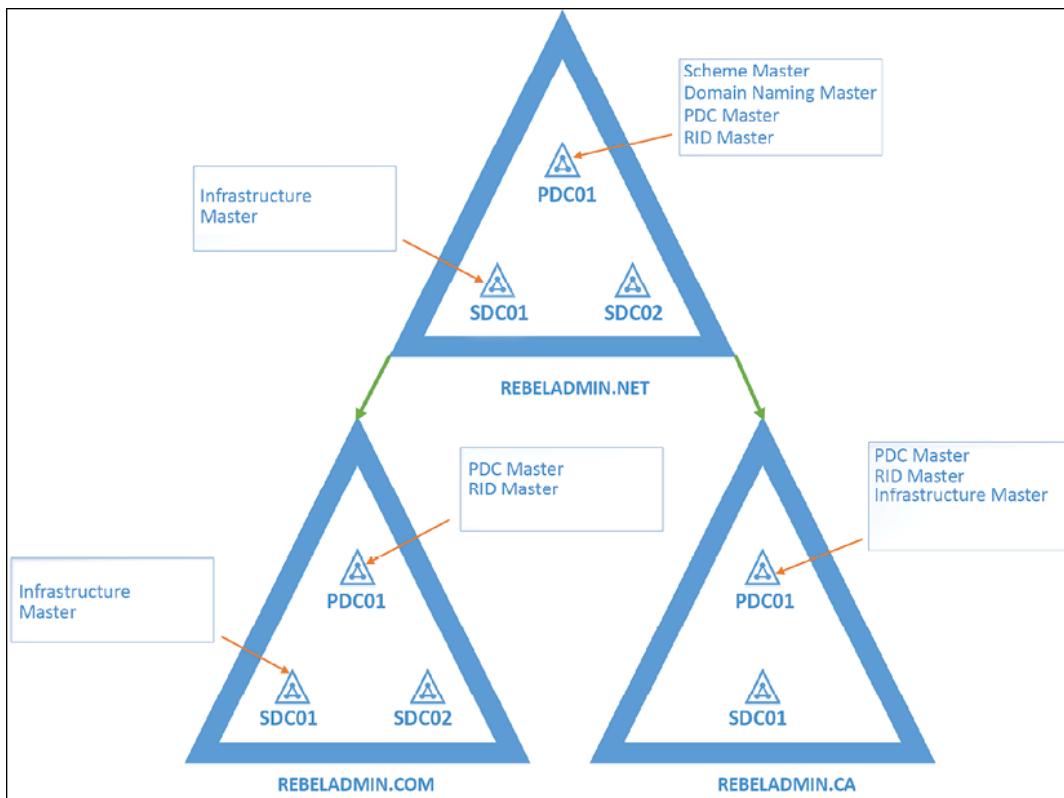


Figure 5.4: FSMO role placement example 2

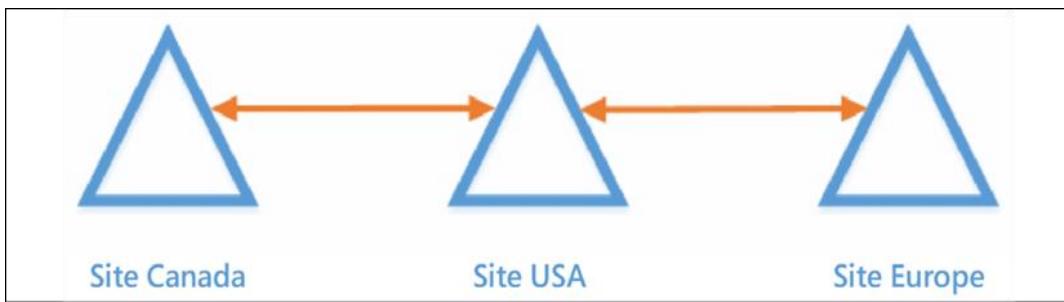


Figure 5.5: Inter-site connectivity

```
PS C:\Users\dfrancis.REBELADMIN> netdom query fsmo
Schema master          REBEL-PDC-01.rebeladmin.com
Domain naming master   REBEL-PDC-01.rebeladmin.com
PDC                   REBEL-SDC02.rebeladmin.com
RID pool manager      REBEL-SDC02.rebeladmin.com
Infrastructure master  REBEL-SDC02.rebeladmin.com
The command completed successfully.
```

```
PS C:\Users\dfrancis.REBELADMIN>
```

Figure 5.6: Migrating the selected number of FSMO roles

```
PS C:\Users\dfrancis.REBELADMIN> netdom query fsmo
Schema master          REBEL-SDC02.rebeladmin.com
Domain naming master   REBEL-SDC02.rebeladmin.com
PDC                   REBEL-SDC02.rebeladmin.com
RID pool manager      REBEL-SDC02.rebeladmin.com
Infrastructure master  REBEL-SDC02.rebeladmin.com
The command completed successfully.
```

Figure 5.7: Migrating FSMO roles

```
PS C:\Users\dfrancis>
PS C:\Users\dfrancis>
PS C:\Users\dfrancis> netdom query fsmo
Schema master          REBEL-SDC02.rebeladmin.com
Domain naming master   REBEL-SDC02.rebeladmin.com
PDC                   REBEL-SDC02.rebeladmin.com
RID pool manager      REBEL-SDC02.rebeladmin.com
Infrastructure master  REBEL-SDC02.rebeladmin.com
The command completed successfully.

PS C:\Users\dfrancis> ping REBEL-SDC02 -t

Pinging rebel-sdc02.rebeladmin.com [10.2.0.5] with 32 bytes of data:
Request timed out.

Ping statistics for 10.2.0.5:
  Packets: Sent = 5, Received = 0, Lost = 5 (100% loss),
Control-C
```

Figure 5.8: Domain controller ping test

```
PS C:\Program Files\PowerShell\7> netdom query fsmo
Schema master          REBEL-PDC-01.rebeladmin.com
Domain naming master   REBEL-PDC-01.rebeladmin.com
PDC                   REBEL-PDC-01.rebeladmin.com
RID pool manager      REBEL-PDC-01.rebeladmin.com
Infrastructure master  REBEL-PDC-01.rebeladmin.com
The command completed successfully.
```

```
PS C:\Program Files\PowerShell\7> -
```

Figure 5.9: Seizing FSMO roles

Commands

Command 5.1

In the Active Directory domain, the PDC role owner can be found using the following command:

```
Get-ADDomain | select PDCEmulator
```

Command 5.2

In the Active Directory domain, the infrastructure operations master role owner can be found using the following command:

```
Get-ADDomain | select InfrastructureMaster
```

Chapter 11

Figures

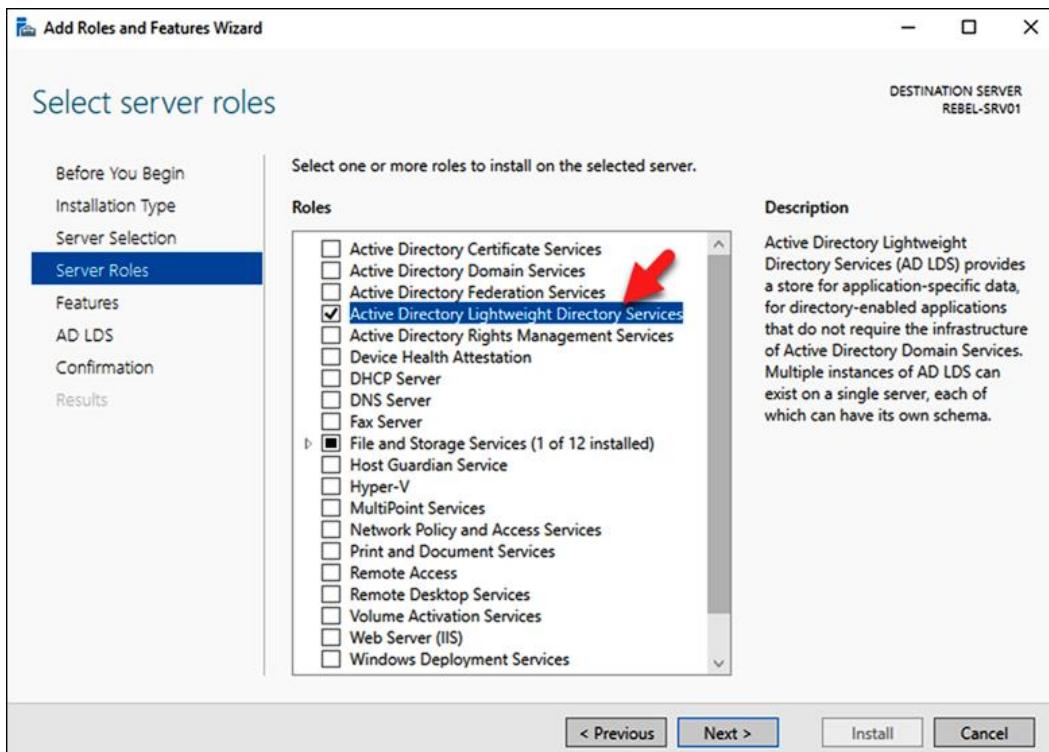


Figure 11.1: AD LDS role

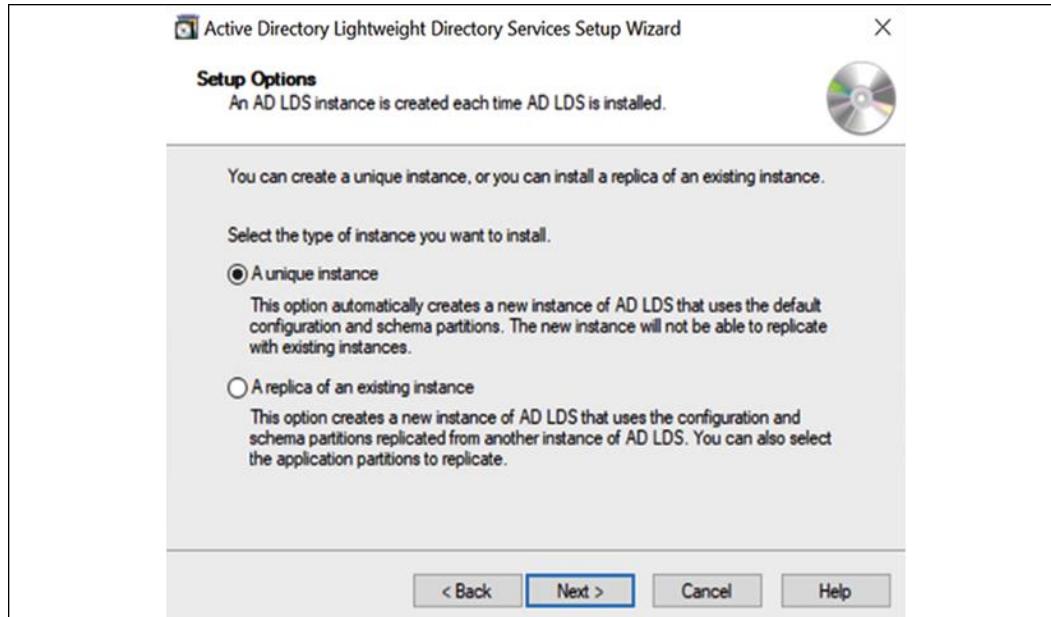


Figure 11.2: AD LDS instance type

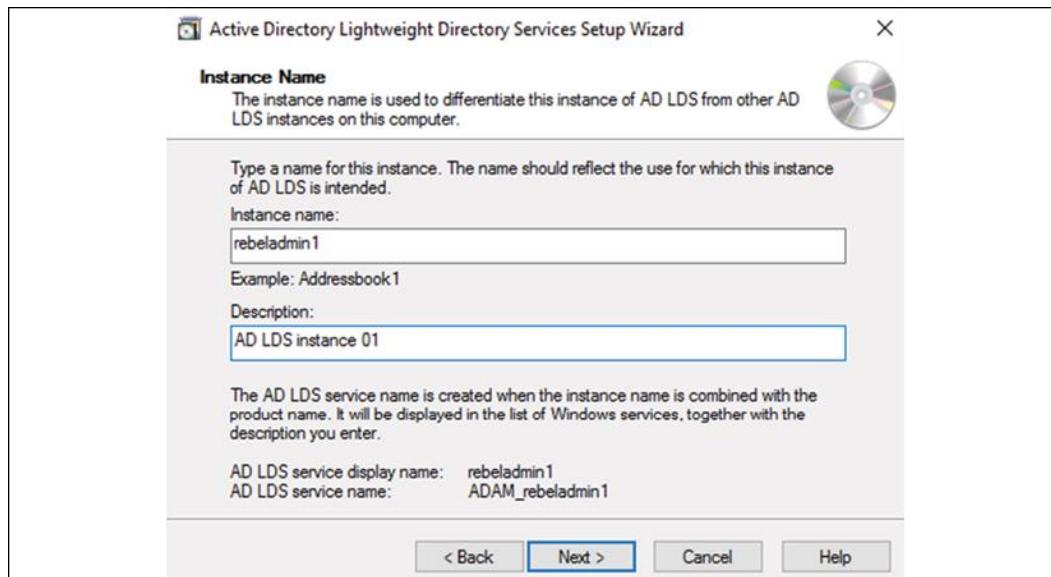


Figure 11.3: AD LDS Instance Name

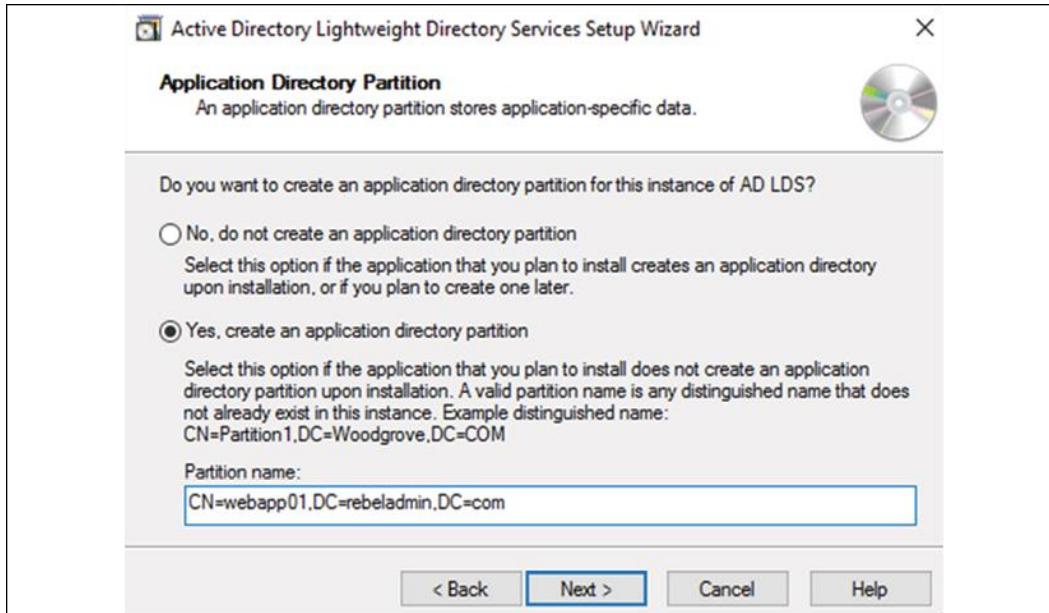


Figure 11.4: AD LDS application directory partition

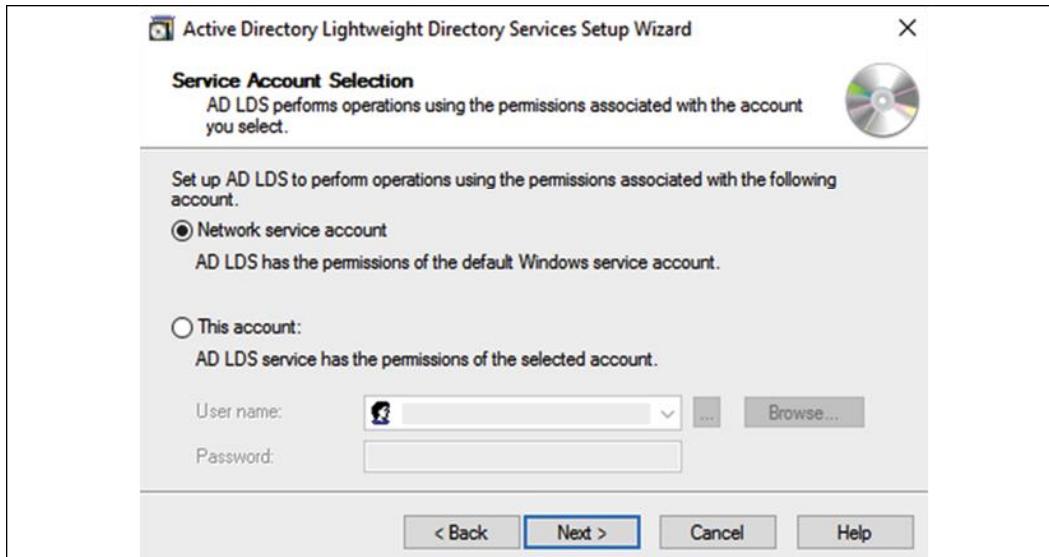


Figure 11.5: AD LDS service account

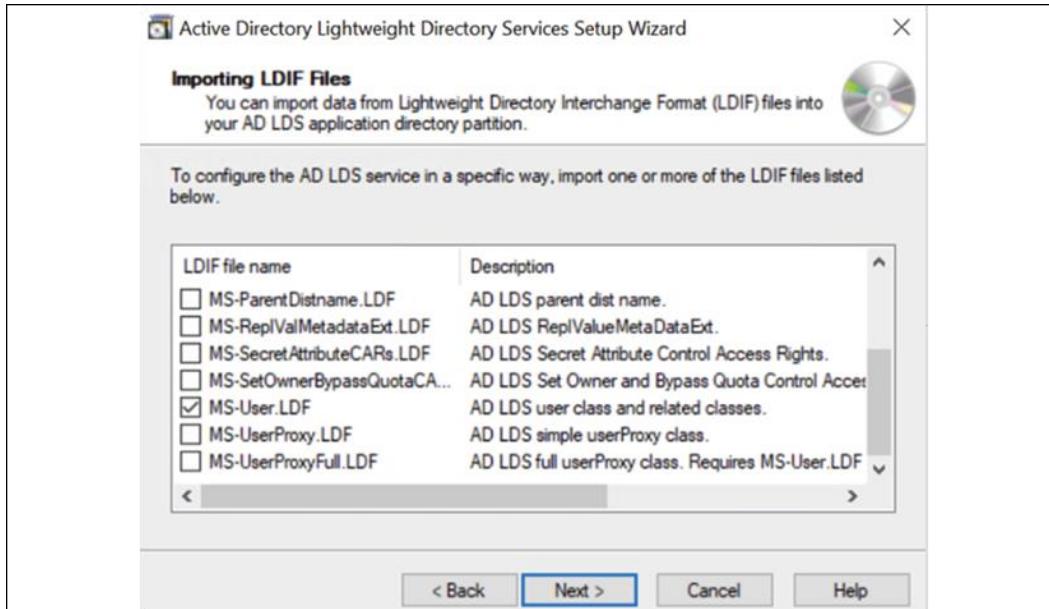


Figure 11.6: AD LDS – Importing LDIF Files

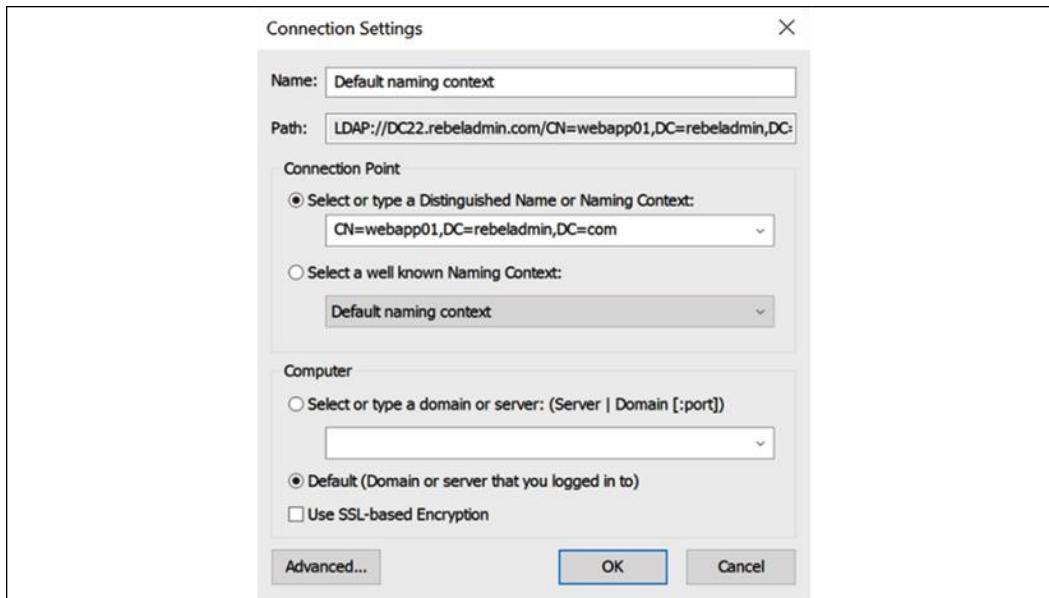


Figure 11.7: AD LDS Connection Settings

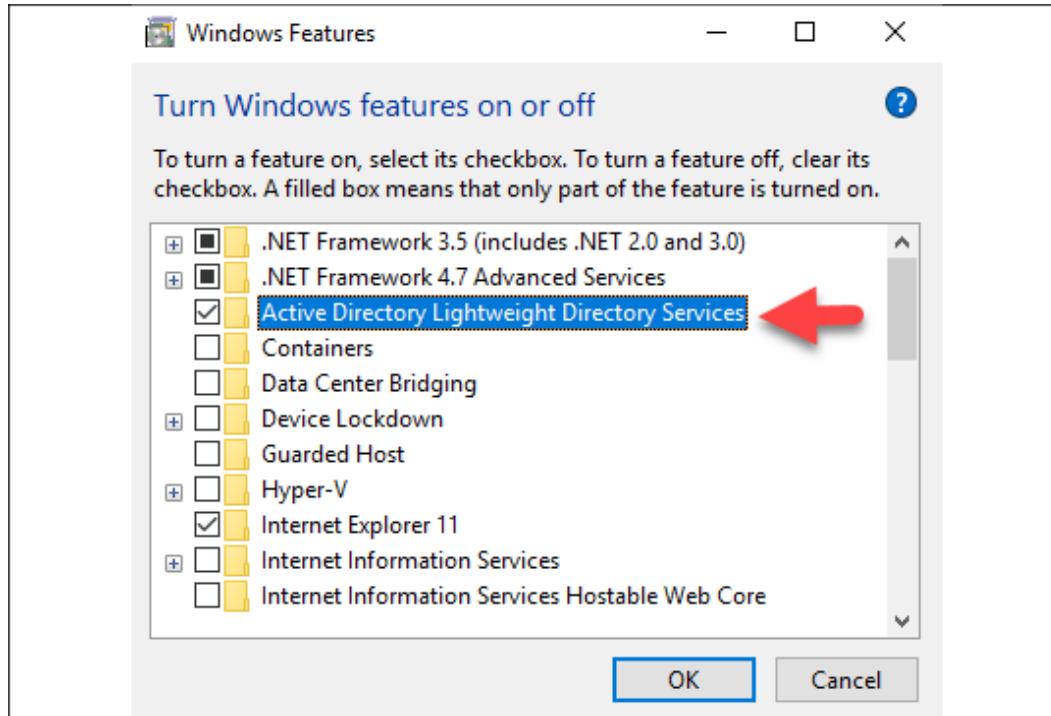


Figure 11.8: AD LDS Windows Features

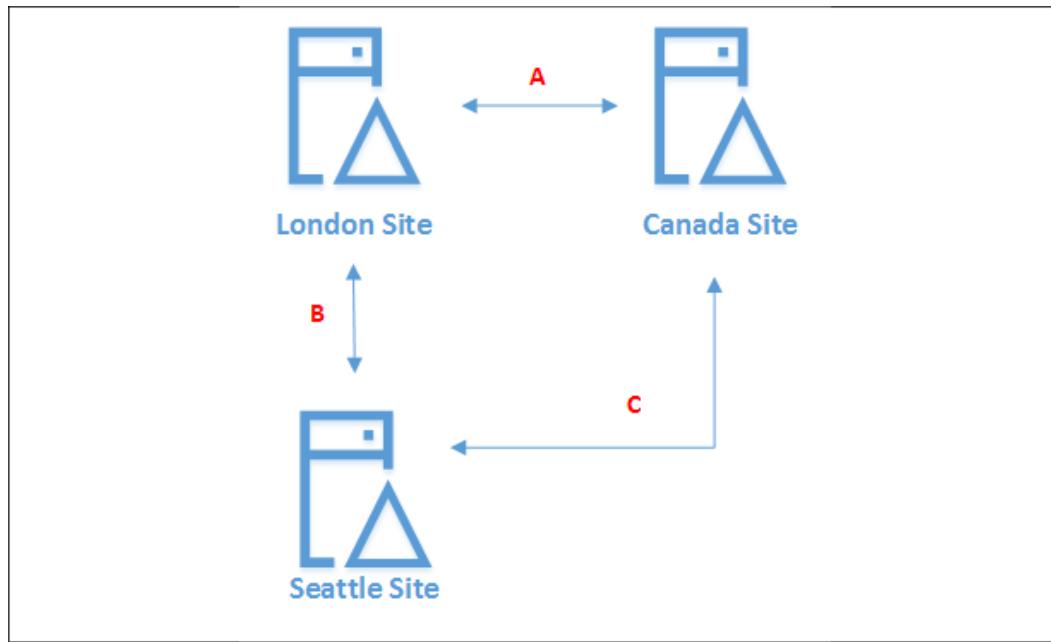


Figure 11.9: Site-link bridges example

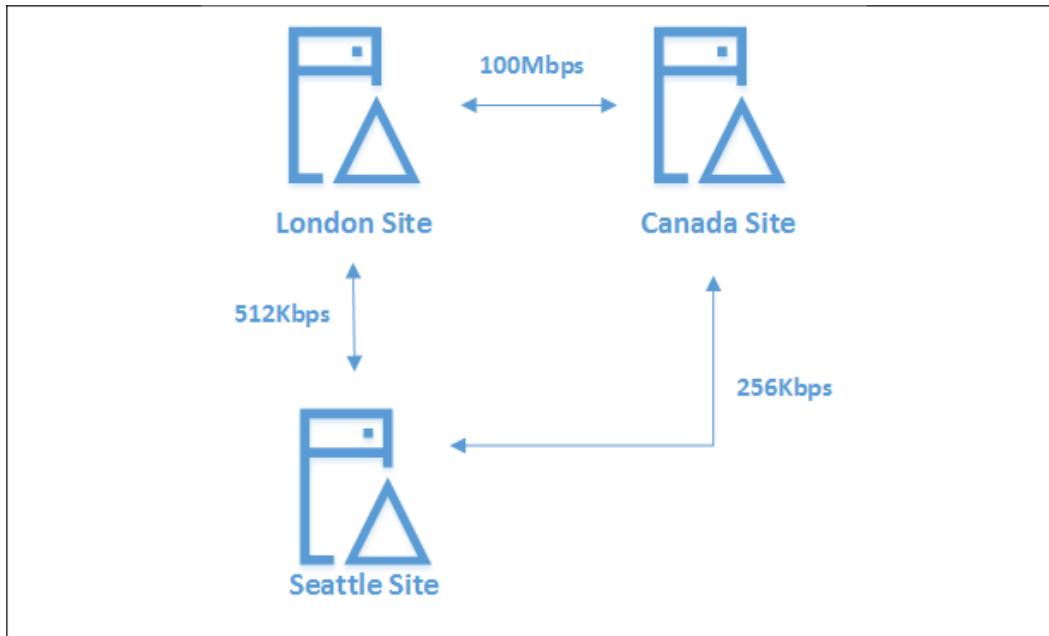


Figure 11.10: Site link cost example

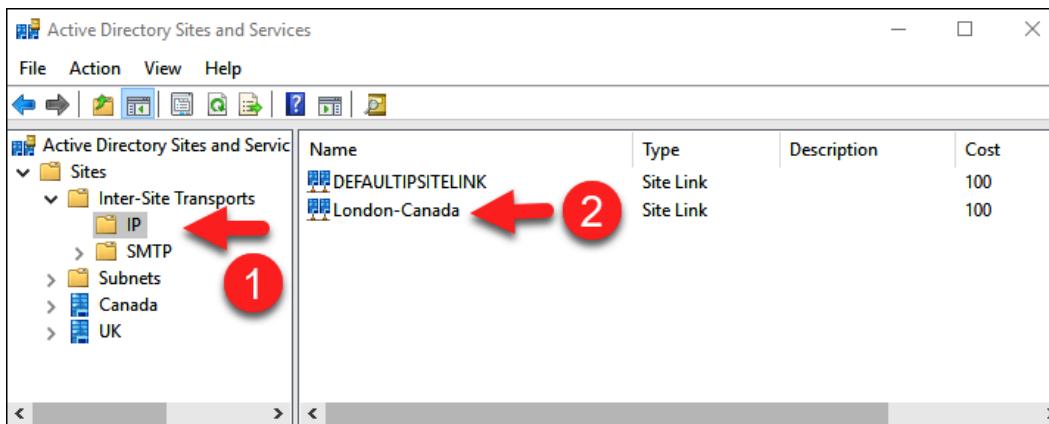


Figure 11.11: Site link

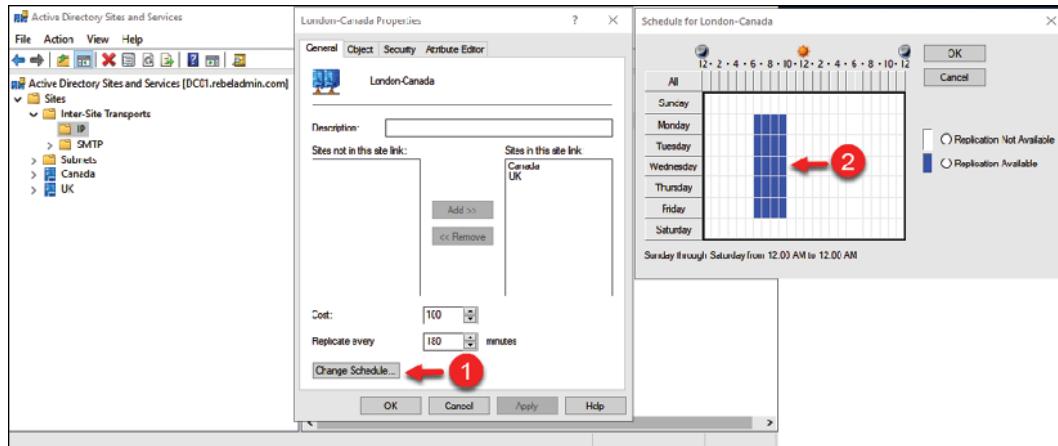


Figure 11.12: Site link replication schedule

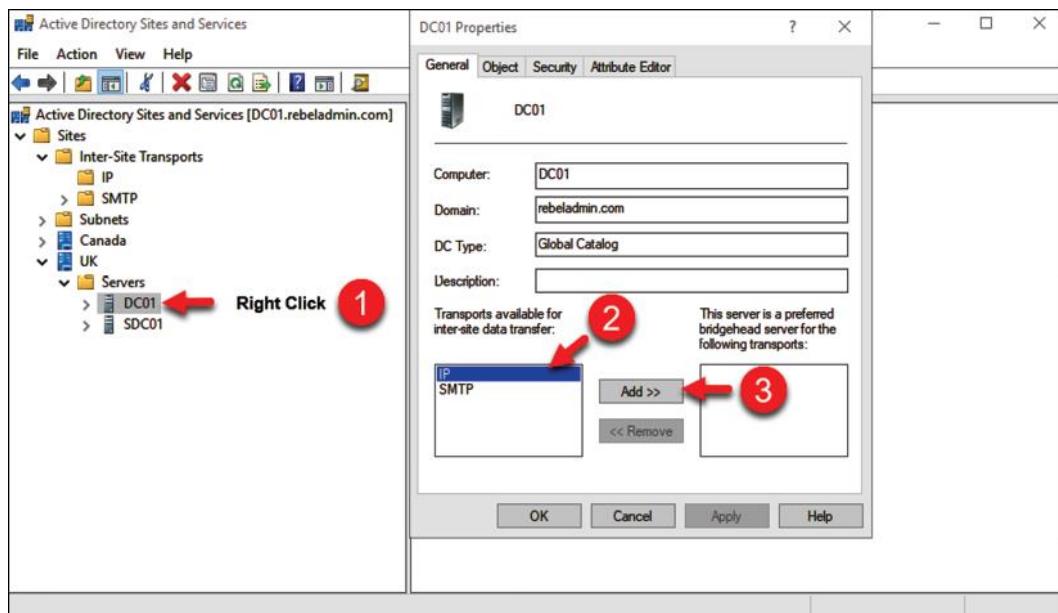


Figure 11.13: Bridgehead server selection

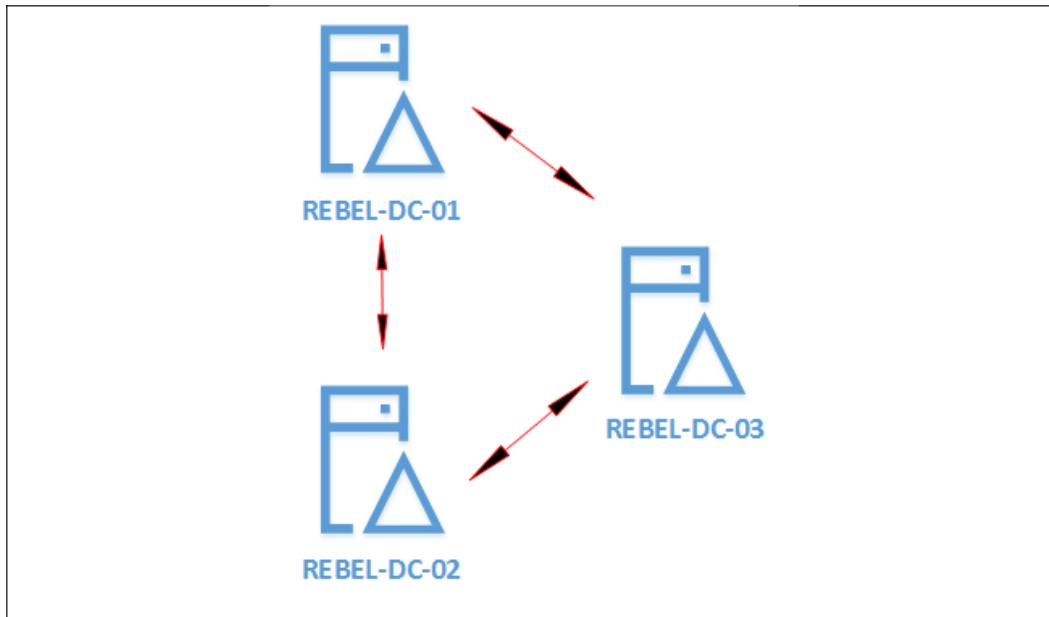


Figure 11.14: Intra-site replication

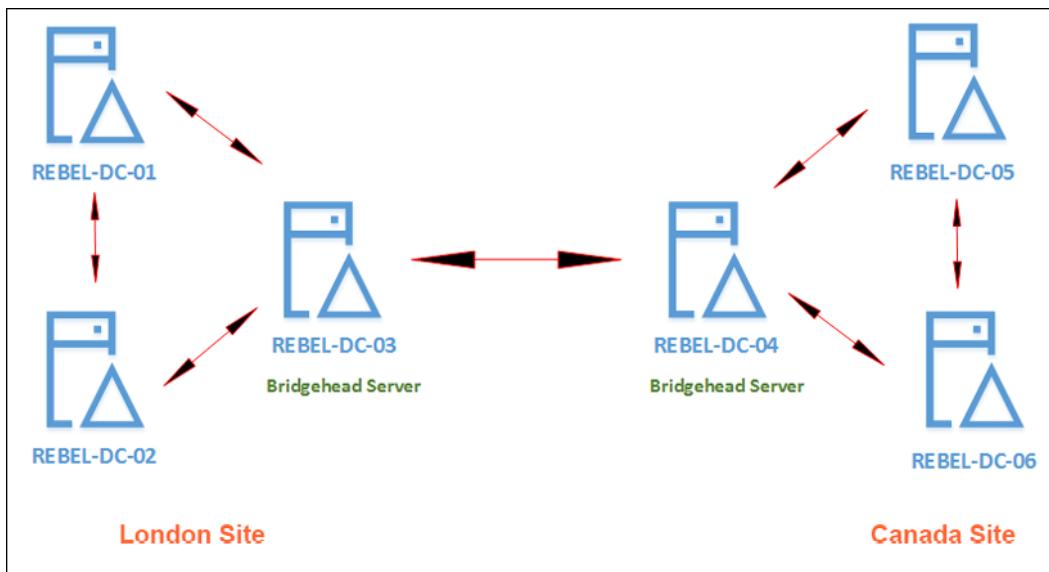


Figure 11.15: Inter-site replication

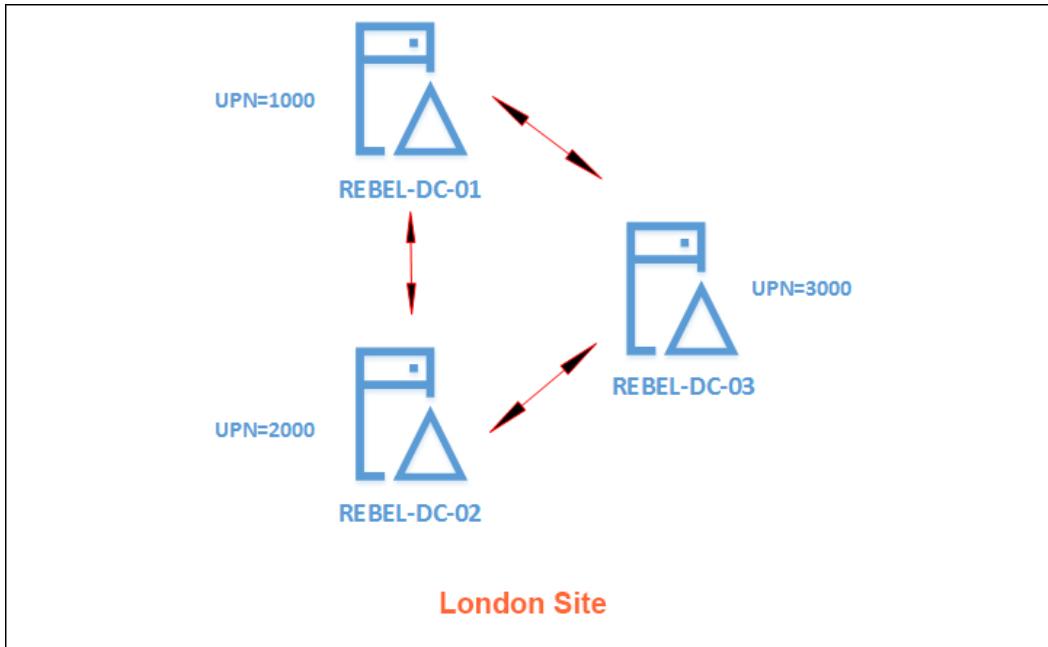


Figure 11.16: UPN value

Commands

The other method is by using PowerShell cmdlets. These are the same commands that we use for AD DS user object management, with the only difference being the need to define the **DN** and server:

```
New-ADUser -name "tidris" -Displayname "Talib Idris" -server
'localhost:389' -path "CN=webapp01,DC=rebeladmin,DC=com"
```

The preceding command creates a user account called tidris on the local LDS instance that runs on 389. Its DNS path is CN=webapp01,DC=rebeladmin,DC=com:

```
Get-ADUser -Filter * -SearchBase "CN=webapp01,DC=rebeladmin,DC=com" -
server 'localhost:389'
```

The preceding command lists all the user accounts in the LDS instance, CN=webapp01,DC=rebeladmin,DC=com. If you'd like to learn about more commands, please refer to *Chapter 7, Managing Active Directory Objects*.

Links

- Windows Server version 1709: <https://bit.ly/30WjqPI>.
- A step-by-step guide for FRS to DFSR migration is available on <https://bit.ly/2Zj1huT>.

Tables

As an example, the logarithm of 512 is 2.709. When we divide 1,024 by 2.709, we get 377.999. So, the link cost of a 512 Kbps line is 378:

Available bandwidth	Cost
9.6 Kbps	1,042
19.2 Kbps	798
38.4 Kbps	644
56 Kbps	586
64 Kbps	567
128 Kbps	486
256 Kbps	425
512 Kbps	378
1,024 Kbps	340
2,048 Kbps	309
4,096 Kbps	283
10 Mbps	257
100 Mbps	205
1,000 Mbps	171

Table 11.1

Managing sites

When the first domain controller is introduced into the infrastructure, the system creates its first site as Default-First-Site-Name. This can be changed based on the requirements. We can review the existing site's configuration using the following PowerShell cmdlet:

```
Get-ADReplicationSite -Filter *
```

It will list the site's information for the AD environment.

Our example only has the default AD site. As this is the first step, we need to change it to a meaningful name so we can assign objects and configurations accordingly. To do that, we can use the Rename-ADObject cmdlet:

```
Rename-ADObject -Identity "CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=rebeladmin,DC=com" -NewName "LondonSite"
```

The preceding command renames the Default-First-Site-Name site to LondonSite. We also can change the site configuration values using the Set-ADReplicationSite cmdlet:

```
Get-ADReplicationSite -Identity LondonSite | Set-ADReplicationSite -Description "UK AD Site"
```

The preceding command changed the site description to UK AD Site.

We can create a new AD site using the New-ADReplicationSite cmdlet. The full description of the command can be viewed using Get-Command New-ADReplicationSite -Syntax:

```
New-ADReplicationSite -Name "CanadaSite" -Description "Canada AD Site"
```

The preceding command creates a new AD site called CanadaSite.

Once the sites are created, we need to move the domain controllers to the relevant sites. By default, all the domain controllers are placed under the default site, Default-First-Site-Name.

In the following command, we are listing all the domain controllers in the AD environment with filtered data to show the Name,ComputerObjectDN,Site attribute values:

```
Get-ADDomainController -Filter * | select Name,ComputerObjectDN,Site | fl
```

Now we have the list of domain controllers; in the next step, we can move the domain controller to the relevant site:

```
Move-ADDirectoryServer -Identity "REBEL-SDC-02" -Site "CanadaSite"
```

The preceding command will move the REBEL-SDC-02 domain controller to CanadaSite.

During the additional domain controller setup, we can define which site it will be assigned to. If the site already has domain controllers, it will do the initial replication from the site local domain controller. If it doesn't, it will replicate from any selected domain controller or, if not, from any available domain

controller. If the link bandwidth is an issue, it's recommended to promote the domain controller from a site that has fast links, and then move the domain controller to the relevant site.

The site link bridge

We can create the site link bridge using the `New-ADReplicationSiteLinkBridge` cmdlet:

```
New-ADReplicationSiteLinkBridge -Name "London-Canada-Bridge" -  
SiteLinksIncluded "London-Canada", "London-CanadaDRLink"
```

The preceding command creates a new site link bridge called London-Canada-Bridge using two site links: London-Canada and London-CanadaDRLink.

Using the `Set-ADReplicationSiteLinkBridge` cmdlet, the existing site link bridge value can change:

```
Set-ADReplicationSiteLinkBridge -Identity "London-Canada-Bridge" -  
SiteLinksIncluded @{Remove='London-CanadaDRLink'}
```

The preceding command removes the London-CanadaDRLink site link from the existing site link bridge, London-Canada-Bridge:

```
Set-ADReplicationSiteLinkBridge -Identity "London-Canada-Bridge" -  
SiteLinksIncluded @{Add='London-CanadaDRLink'}
```

The preceding command adds the given site link to the existing site link bridge.

Chapter 12

Figures



Figure 12.1: Direction of Trust vs Direction of Access

```
PS C:\Users\dfrancis> ping contoso.com

Pinging contoso.com [40.113.200.201] with 32 bytes of data:
Request timed out.

Ping statistics for 40.113.200.201:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PS C:\Users\dfrancis> _
```

Figure 12.2: Ping test to check DNS

```
PS C:\Users\dfrancis> hostname
DC01
PS C:\Users\dfrancis> ping contoso.com

Pinging contoso.com [10.1.5.4] with 32 bytes of data:
Reply from 10.1.5.4: bytes=32 time=1ms TTL=128
Reply from 10.1.5.4: bytes=32 time=1ms TTL=128
Reply from 10.1.5.4: bytes=32 time=2ms TTL=128
Reply from 10.1.5.4: bytes=32 time=1ms TTL=128

Ping statistics for 10.1.5.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms
PS C:\Users\dfrancis> _
```

Figure 12.3: Ping result after conditional forwarder setup – contoso.com

```
Administrator: C:\Program Files\PowerShell\7\pwsh.exe
PS C:\Windows\System32> Add-DnsServerConditionalForwarderZone -Name "rebeladmin.com" -ReplicationScope "Forest" -MasterServers 10.1.0.4
PS C:\Windows\System32> ipconfig /flushdns

Windows IP Configuration

Successfully flushed the DNS Resolver Cache.
PS C:\Windows\System32> ping rebeladmin.com

Pinging rebeladmin.com [10.1.0.4] with 32 bytes of data:
Reply from 10.1.0.4: bytes=32 time=1ms TTL=128
Reply from 10.1.0.4: bytes=32 time=2ms TTL=128
Reply from 10.1.0.4: bytes=32 time=1ms TTL=128
Reply from 10.1.0.4: bytes=32 time=1ms TTL=128

Ping statistics for 10.1.0.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 2ms, Average = 1ms
PS C:\Windows\System32>
```

Figure 12.4: Ping result after conditional forwarder setup – rebeladmin.com

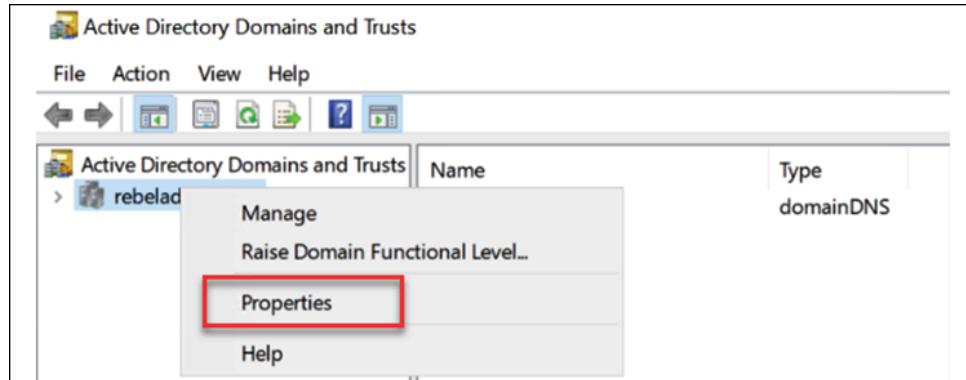


Figure 12.5: Domain Properties

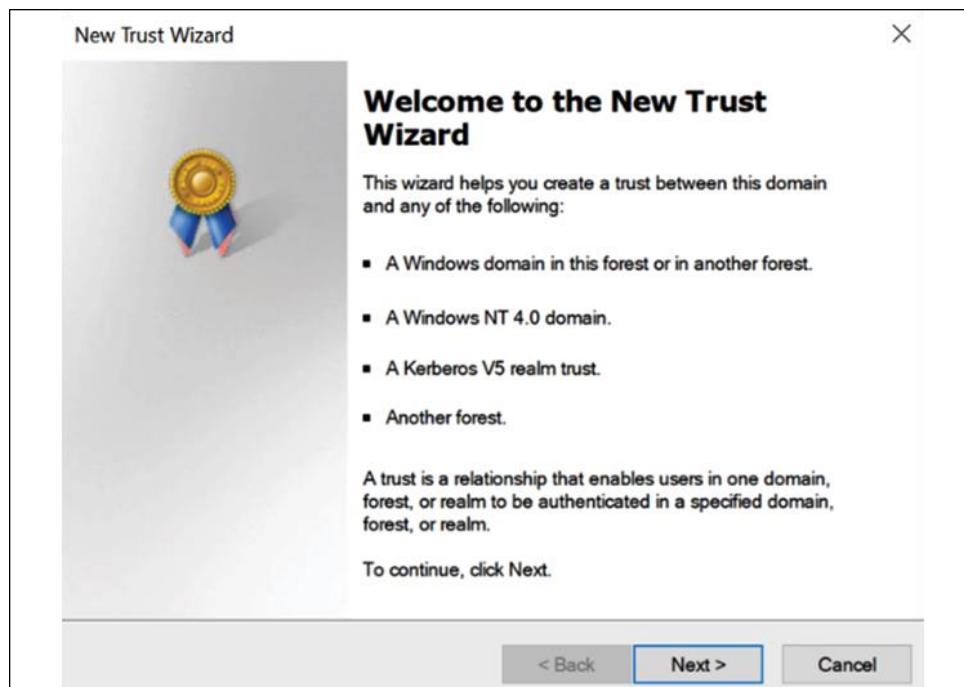


Figure 12.6: New Trust Wizard

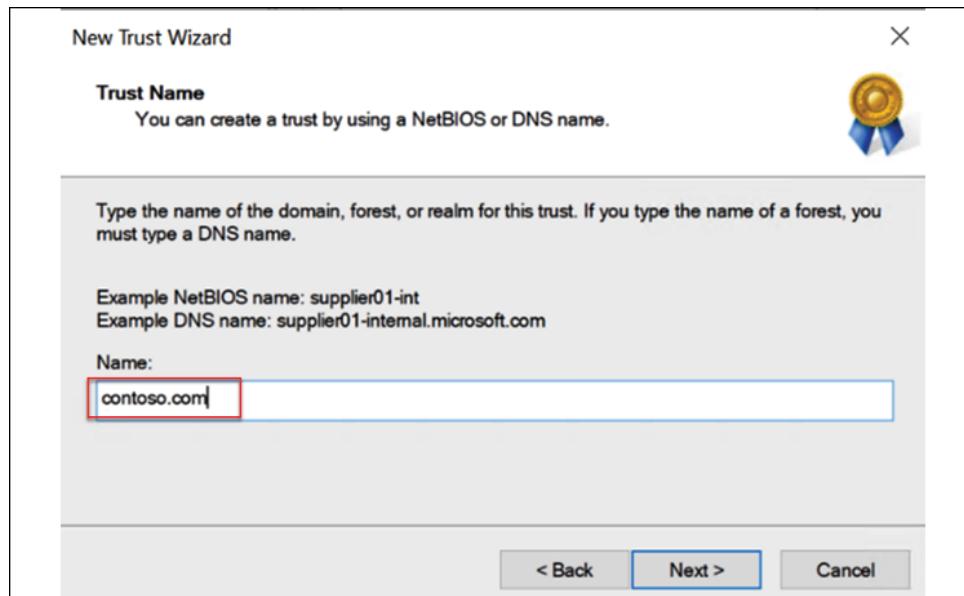


Figure 12.7: Remote Domain Name

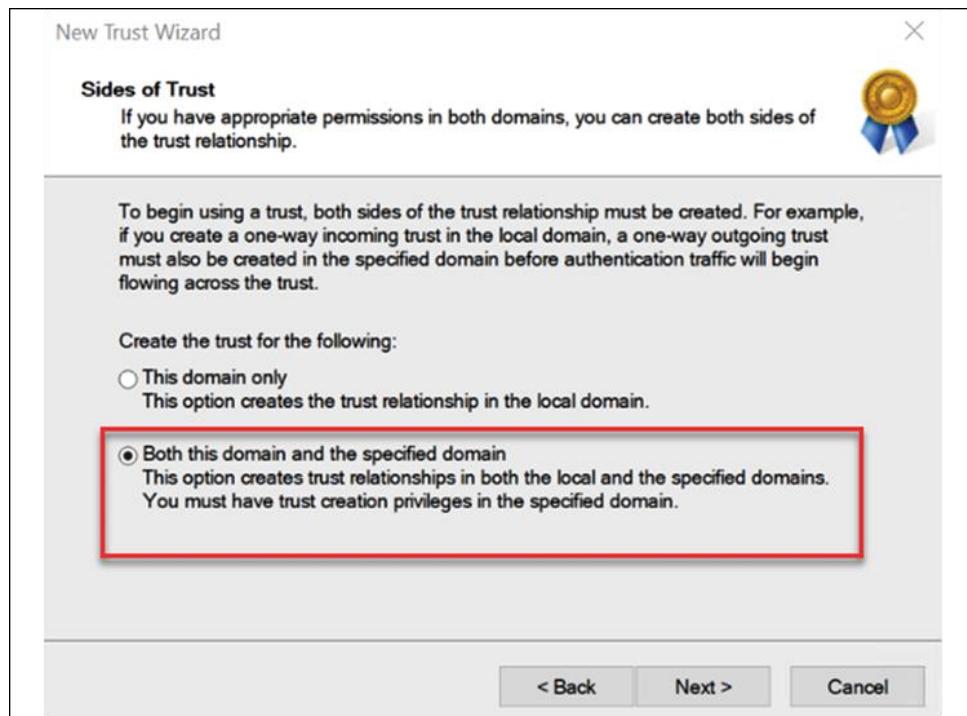


Figure 12.8: Create a trust on both sides

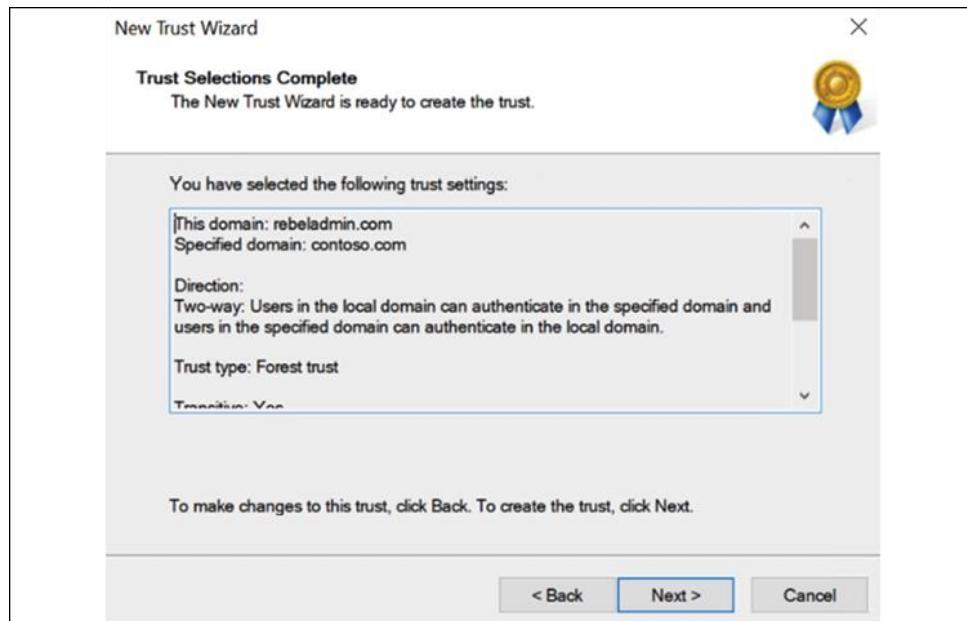


Figure 12.9: Verify Trust configuration

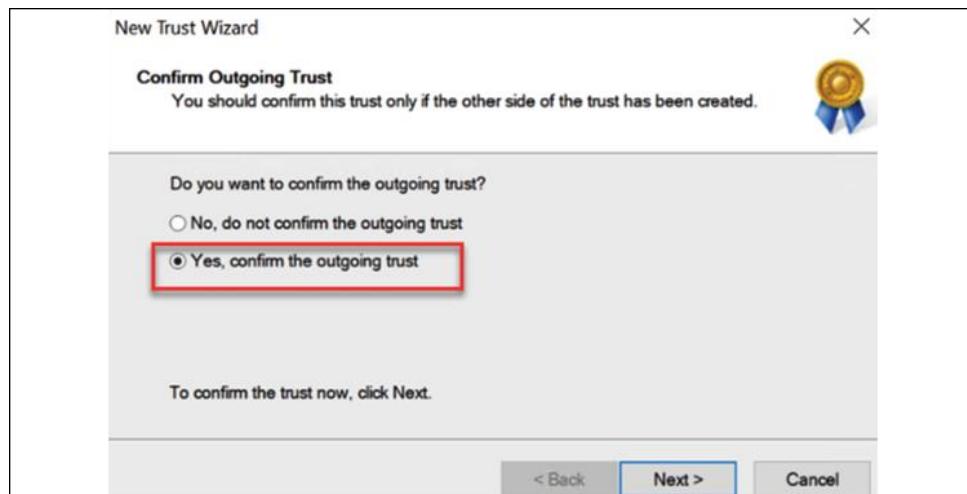


Figure 12.10: Confirm outgoing trust

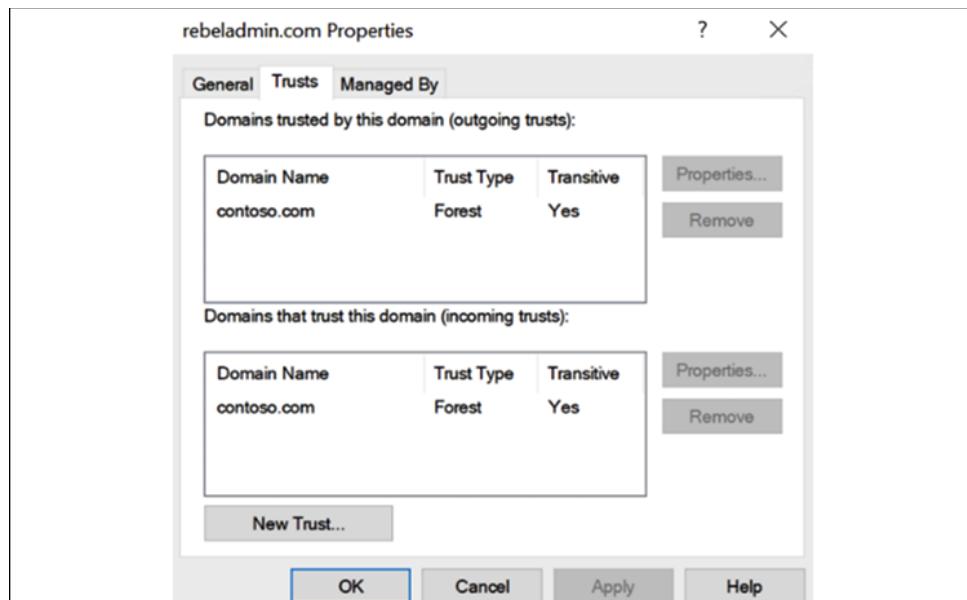


Figure 12.11: Trust details – rebeladmin.com

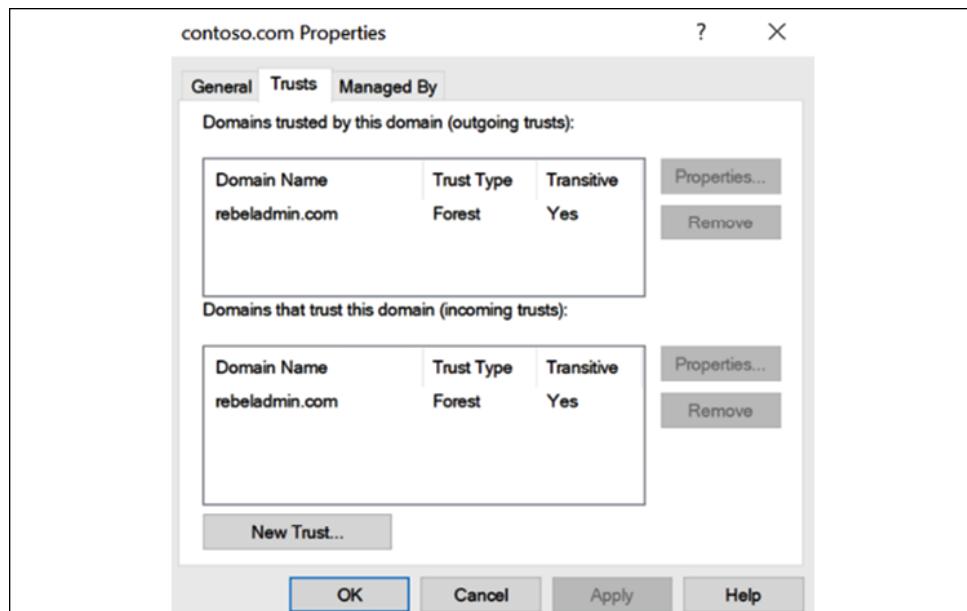


Figure 12.12: Trust details – contoso.com

```

Administrator: C:\Program Files\PowerShell\7\pwsh.exe
PowerShell 7.1.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Windows\System32> Get-ADUser -Server CON-DC01.contoso.com -Filter * -SearchBase "OU=Test,DC=CONTOSO,DC=COM"
DistinguishedName : CN=usera,OU=Test,DC=contoso,DC=com
Enabled          : True
GivenName        : usera
Name             : usera
ObjectClass     : user
ObjectGUID       : 198e7347-9362-481f-a82d-15d0d4445167
SamAccountName   : usera
SID              : S-1-5-21-1797065658-1832406859-3879835726-1602
Surname          :
UserPrincipalName : usera@contoso.com

PS C:\Windows\System32>

```

Figure 12.13: Active Directory User Query – contoso.com

```

PS C:\Windows\System32> Get-ADUser -Server DC01.rebeladmin.com -Filter * -SearchBase "OU=Sales,DC=rebeladmin,DC=com"
DistinguishedName : CN=salesa,OU=Sales,DC=rebeladmin,DC=com
Enabled          : True
GivenName        : salesa
Name             : salesa
ObjectClass     : user
ObjectGUID       : 814ffd11-b6a6-486f-91aa-2ef798f976c1
SamAccountName   : salesa
SID              : S-1-5-21-4070376937-2589962512-1042476643-3601
Surname          :
UserPrincipalName : salesa@rebeladmin.com

```

Figure 12.14: Active Directory User Query – rebeladmin.com

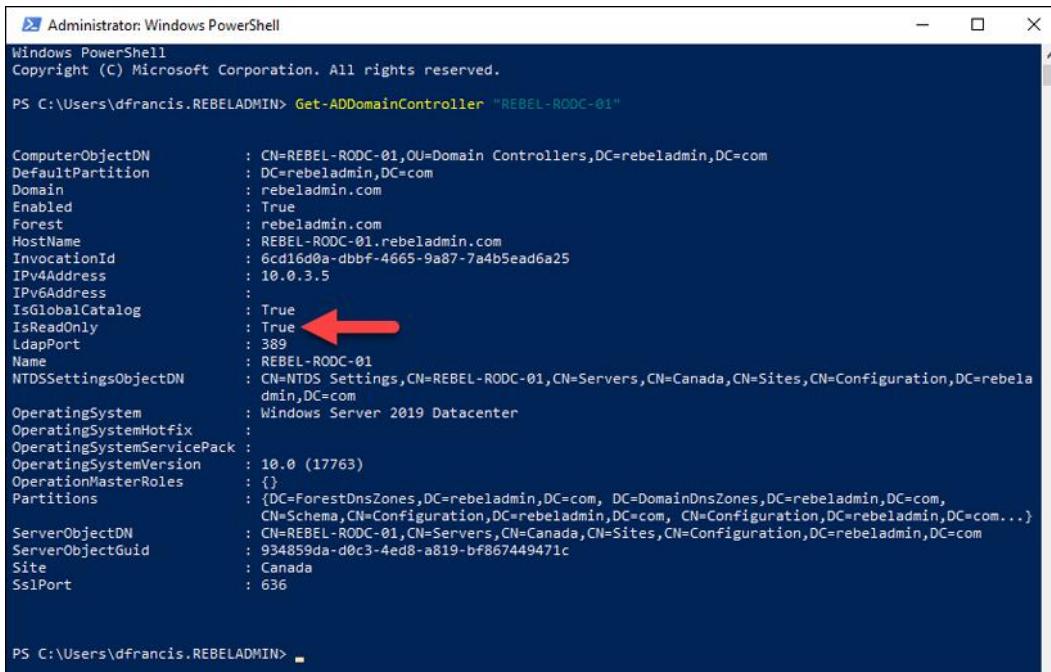
```

PS C:\Windows\System32> hostname
DC01
PS C:\Windows\System32> Get-ADDomain contoso.com

AllowedDNSSuffixes          : {}
ChildDomains                 : {}
ComputersContainer           : CN=Computers,DC=contoso,DC=com
DeletedObjectsContainer      : CN=Deleted Objects,DC=contoso,DC=com
DistinguishedName            : DC=contoso,DC=com
DNSRoot                      : contoso.com
DomainControllersContainer   : OU=Domain Controllers,DC=contoso,DC=com
DomainMode                   : Windows2016Domain
DomainSID                    : S-1-5-21-1797065658-1832406859-3879835726
ForeignSecurityPrincipalsContainer : CN=ForeignSecurityPrincipals,DC=contoso,DC=com
Forest                       : contoso.com
InfrastructureMaster          : CON-DC01.contoso.com
LastLogonReplicationInterval : 
LinkedGroupPolicyObjects     : {CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=contoso,DC=com}
LostAndFoundContainer         : CN=LostAndFound,DC=contoso,DC=com
ManagedBy                     : 
Name                          : contoso
NetBIOSName                  : CONTOSO
ObjectClass                  : domainDNS
ObjectGUID                   : bcb8dfca-26c5-4a54-bf16-37ef2e282473
ParentDomain                 : 
PDCEmulator                  : CON-DC01.contoso.com
PublicKeyRequiredPasswordRolling : True
QuotasContainer               : CN=NTDS Quotas,DC=contoso,DC=com
ReadOnlyReplicaDirectoryServers : {}
ReplicaDirectoryServers       : {CON-DC01.contoso.com}
RIDMaster                     : CON-DC01.contoso.com
SubordinateReferences         : {DC=ForestDnsZones,DC=contoso,DC=com, DC=DomainDnsZones,DC=contoso,DC=com, CN=Configuration,DC=contoso,DC=com}
SystemsContainer               : CN=System,DC=contoso,DC=com
UsersContainer                : CN=Users,DC=contoso,DC=com

```

Figure 12.15: Active Directory Domain Query – contoso.com



```

Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\dfrancis.REBELADMIN> Get-ADDomainController "REBEL-RODC-01"

ComputerObjectDN          : CN=REBEL-RODC-01,OU=Domain Controllers,DC=rebeladmin,DC=com
DefaultPartition           : DC=rebeladmin,DC=com
Domain                     : rebeladmin.com
Enabled                   : True
Forest                     : rebeladmin.com
HostName                  : REBEL-RODC-01.rebeladmin.com
InvocationId               : 6cd16d0a-dbbf-4665-9a87-7a4b5ead6a25
IPv4Address                : 10.0.3.5
IPv6Address                :
IsGlobalCatalog            :
IsReadOnly                 : True ← Red arrow points here
LdapPort                   : 389
Name                       : REBEL-RODC-01
NTDSSettingsObjectDN      : CN=NTDS Settings,CN=REBEL-RODC-01,CN=Servers,CN=Canada,CN=Sites,CN=Configuration,DC=rebeladmin,DC=com
OperatingSystem             : Windows Server 2019 Datacenter
OperatingSystemHotfix        :
OperatingSystemServicePack  :
OperatingSystemServiceVersion : 10.0 (17763)
OperationMasterRoles        : {}
Partitions                 : {DC=ForestDnsZones,DC=rebeladmin,DC=com, DC=DomainDnsZones,DC=rebeladmin,DC=com, CN=Schema,CN=Configuration,DC=rebeladmin,DC=com, CN=Configuration,DC=rebeladmin,DC=com...}
ServerObjectDN              : CN=REBEL-RODC-01,CN=Servers,CN=Canada,CN=Sites,CN=Configuration,DC=rebeladmin,DC=com
ServerObjectGuid            : 934859da-d0c3-4ed8-a819-bf867449471c
Site                        : Canada
SslPort                     : 636

PS C:\Users\dfrancis.REBELADMIN>

```

Figure 12.16: RODC Settings

Name	Date modified	Type	Size
edb.chk	9/8/2021 9:58 PM	Recovered File Fragm...	8 KB
edb	9/9/2021 9:39 PM	Text Document	10,240 KB
edb00001	9/5/2021 9:37 PM	Text Document	10,240 KB
edbres00001.jrs	9/5/2021 9:37 PM	JRS File	10,240 KB
edbres00002.jrs	9/5/2021 9:37 PM	JRS File	10,240 KB
edbtmp	9/5/2021 9:37 PM	Text Document	10,240 KB
ntds.dit	9/9/2021 9:16 PM	DIT File	16,384 KB
ntds.jfm	9/9/2021 9:16 PM	JFM File	16 KB
temp.edb	9/9/2021 9:16 PM	EDB File	424 KB

Figure 12.17: NTDS Folder

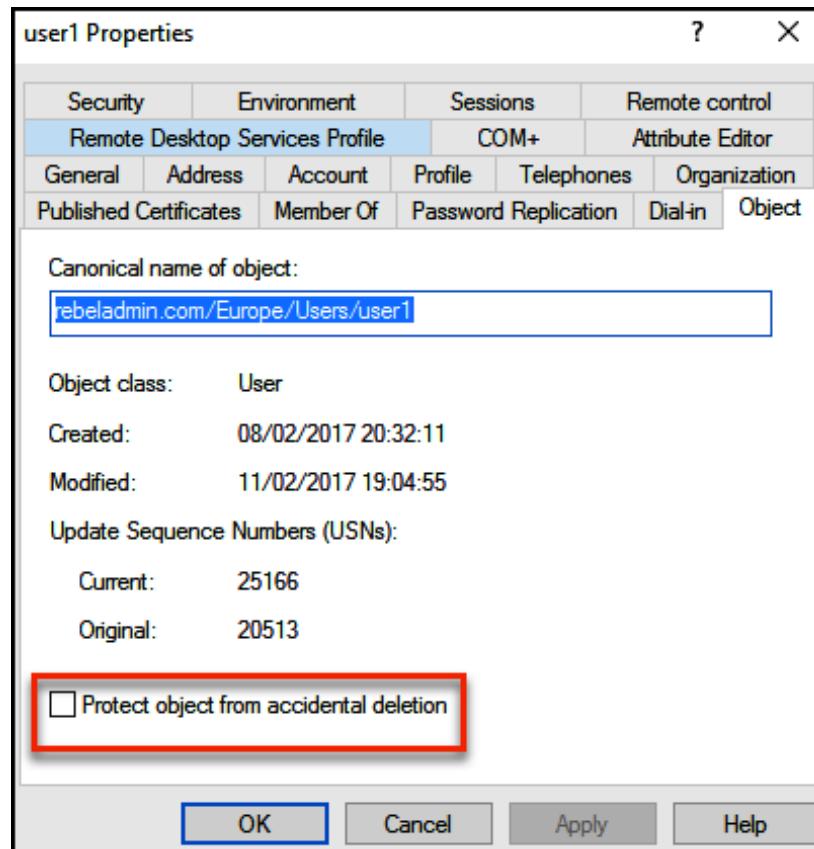


Figure 12.18: Protect objects from accidental deletion



Figure 12.19: Error Message – Protected Object

```
PS C:\Windows\System32> Get-ADObject -IncludeDeletedObjects -Filter {samAccountName -eq 'user01'}
```

Deleted	: True
DistinguishedName	: CN=user01\0ADEL:8d7b204e-e1c9-4b47-9b67-2d4798710186,CN=Deleted Objects,DC=rebeladmin,DC=com
Name	: user01
ObjectClass	: user
ObjectGUID	: 8d7b204e-e1c9-4b47-9b67-2d4798710186

Figure 12.20: Deleted Active Directory Object

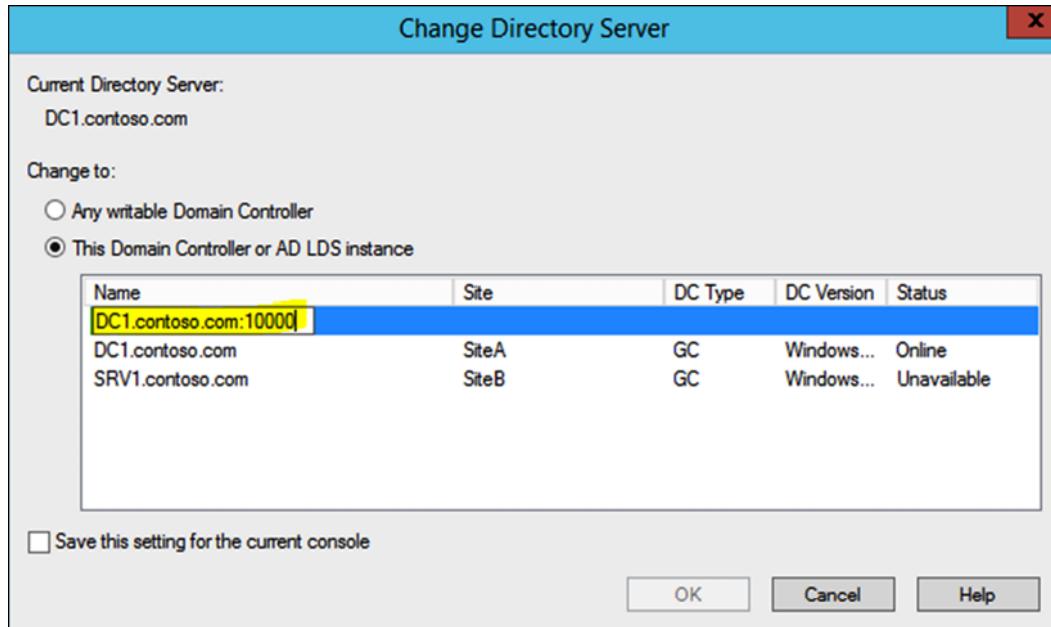


Figure 12.21: Connecting to Snapshot

Tables

Service	Ports
LDAP	TCP 389
LDAPS (SSL)	TCP 636
DNS	TCP/UDP 53
RPC	TCP 135 TCP 1024-65535
SMB	TCP 445
Kerberos	TCP/UDP 88
Global Catalog	TCP 3268
Global Catalog (SSL)	TCP 3269

Table 12.1 – Firewall ports

Domain	Domain Controller	IP Address
rebeladmin.com	DC01.rebeladmin.com	10.1.0.4/24
contoso.com	CON-DC01.contoso.com	10.1.5.4/24

Table 12.2 – IP address configuration of forests in domain environment

Commands

Command 12.1

```
Add-DnsServerConditionalForwarderZone -Name "contoso.com" -  
ReplicationScope "Forest" -MasterServers 10.1.5.4
```

Command 12.2

```
Add-DnsServerConditionalForwarderZone -Name "rebeladmin.com" -  
ReplicationScope "Forest" -MasterServers 10.1.0.4
```

Command 12.3

```
Get-ADUser -Server CON-DC01.contoso.com -Filter * -SearchBase  
"OU=Test,DC=CONTOSO,DC=COM"
```

Command 12.4

```
Get-ADDomain contoso.com
```

Command 12.5

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

Command 12.6

```
Import-Module ADDSDeployment  
Install-ADDSDomainController `  
-Credential (Get-Credential) `  
-CriticalReplicationOnly:$false `  
-DatabasePath "C:\Windows\NTDS" `  
-DomainName "rebeladmin.com" `  
-LogPath "C:\Windows\NTDS" `  
-ReplicationSourceDC "REBEL-PDC-01.rebeladmin.com" `  
-SYSVOLPath "C:\Windows\SYSVOL" `  
-UseExistingAccount:$true `  
-Norebootoncompletion:$false  
-Force:$true
```

Command 12.7

```
Get-ADDomainControllerPasswordReplicationPolicy -Identity REBEL-RODC-  
01 -Allowed
```

Command 12.8

```
Get-ADDomainControllerPasswordReplicationPolicy -Identity REBEL-RODC-  
01 -Denied
```

Command 12.9

```
Add-ADDomainControllerPasswordReplicationPolicy -Identity REBEL-RODC-  
01 -AllowedList "user1"
```

Command 12.10

The following command will add the user object named user2 to the Denied list:

```
Add-ADDomainControllerPasswordReplicationPolicy -Identity REBEL-RODC-  
01 -DeniedList "user2"
```

Command 12.11

The database and log files cannot be moved while AD DS is running. Therefore, the first step of the action is to stop the service:

```
net stop ntds
```

Command 12.12

In my demonstration, I will move it to a folder called ADDB in a different partition:

```
ntdsutil
activate instance ntds
files
move db to E:\ADDB
move logs to E:\ADDB
integrityquit
quit
```

Command 12.13

Once it's completed, we need to start AD DS using the following command:

```
net start ntds
```

Command 12.14

Once the service stops (net stop ntds), we can run the defragmentation using the following commands:

```
Ntdsutil
activate instance ntds
files
compact to E:\CompactDB
quit
quit
```

Command 12.15

Once this is completed, the compact database should be copied to the original ntds.dit location. This can be done by using the following command:

```
copy "E:\CompactDB\ntds.dit" "E:\ADDB\ntds.dit"
```

After that, we also need to delete the old log file:

```
del E:\ADDB\*.log
```

Command 12.16

The AD Recycle Bin feature requires a minimum of a Windows Server 2008 R2 domain and a forest functional level. Once this feature is enabled, it cannot be disabled. This feature can be enabled using the following command:

```
Enable-ADOptionalFeature 'Recycle Bin Feature' -Scope  
ForestOrConfigurationSet -Target rebeladmin.com
```

In the preceding command, -Target can be changed with your domain name.

Command 12.17

Once Recycle Bin Feature is enabled, we can revive the objects that have been deleted using the following command:

```
Get-ADObject -filter 'isdeleted -eq $true' -includeDeletedObjects
```

The preceding command searches for the objects where the isdeleted attributes are set to true.

Command 12.18

Now, we know the deleted object and it can be restored using the following command:

```
Get-ADObject -Filter 'samaccountname -eq "dfrancis"' -  
IncludeDeletedObjects | Restore-ADObject
```

The preceding command restores the user object, dfrancis.

Command 12.19

We can create the AD DS snapshot using ntdsutil. In order to run this, we need to have domain administrator privileges:

```
Ntdsutil  
Snapshot  
activate instance ntds  
create  
quit  
quit
```

Command 12.20

Now, we have a snapshot, and at a later time, it can be mounted. To mount it, we need to use the following command:

```
Ntdsutil  
Snapshot  
activate instance ntds  
list all  
mount 1  
quit  
quit
```

The preceding command mounts a snapshot called 1 from the list, which is listed under the given mount points.

Command 12.21

The next step is to mount the snapshot, which can be done using the following command:

```
dsamain -dbpath C:$SNAP_201703152333_VOLUMEE$ADDBntds.dit -ldapport  
10000
```

In the preceding command, `-dbpath` defines the AD DS database path, and `-ldapport` defines the port used for the snapshot. It can be any available TCP port.

Command 12.22

Once the snapshot is mounted, we can connect to it using the server's name and the LDAP port, 10000. Once the work is finished, it needs to be unmounted as well. To do that, we can use the following command:

```
Ntdsutil  
Snapshot  
activate instance ntds  
list all  
unmount 1  
quit  
quit
```

Command 12.23

The first step to proceed with configuration is to install the Windows backup feature in the Active Directory server:

```
Install-WindowsFeature -Name Windows-Server-Backup -  
IncludeAllSubFeature
```

Command 12.24

Next, let's create a backup policy using the following command:

```
$BKPolicy = New-WBPolicy
```

Command 12.25

Then, let's go ahead and add a system state to the policy:

```
Add-WBSystemState -Policy $BKPolicy
```

Command 12.26

It also needs the backup volume path:

```
$Bkpath = New-WBBackupTarget -VolumePath "F:"
```

Command 12.27

Now, we need to map the policy with the path:

```
Add-WBBackupTarget -Policy $BKPolicy -Target $Bkpath
```

Command 12.28

Finally, we can run the backup using the following command:

```
Start-WBBackup -Policy $BKPolicy
```

Command 12.29

Once it's loaded in safe mode, we can use the following commands:

```
$ADBackup = Get-WBBackupSet | select -Last 1 Start-WBSystemStateRecovery -BackupSet $ADBackup
```

This will restore the most recent backup the system has taken.

Chapter 13

Figures

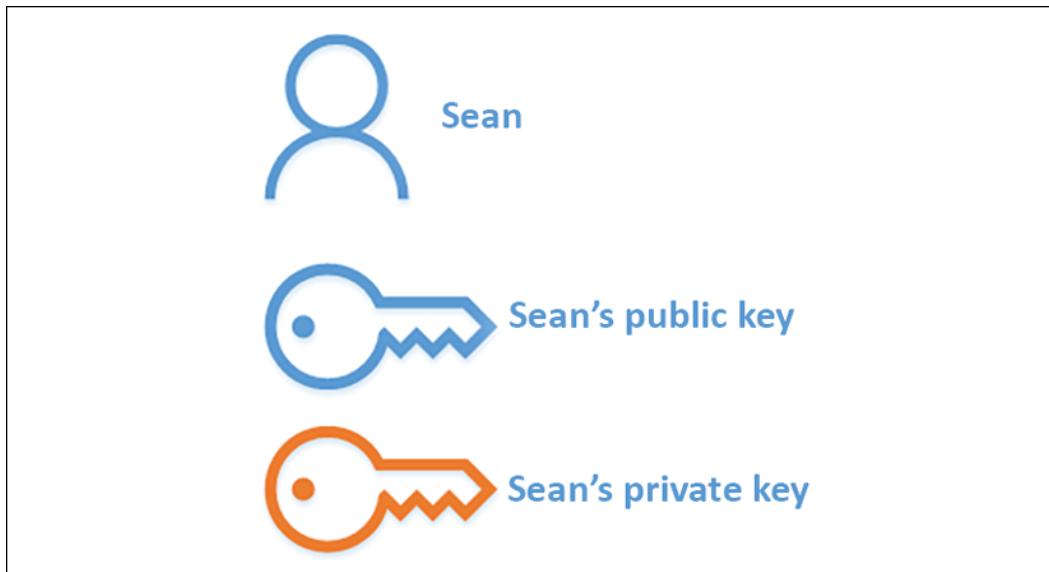


Figure 13.1: A user's key pair

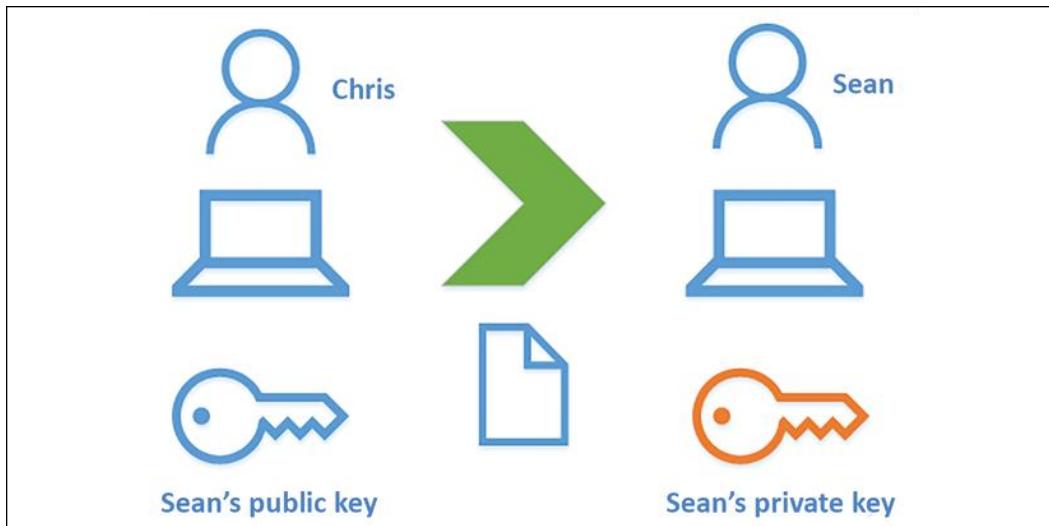


Figure 13.2: Digital encryption example

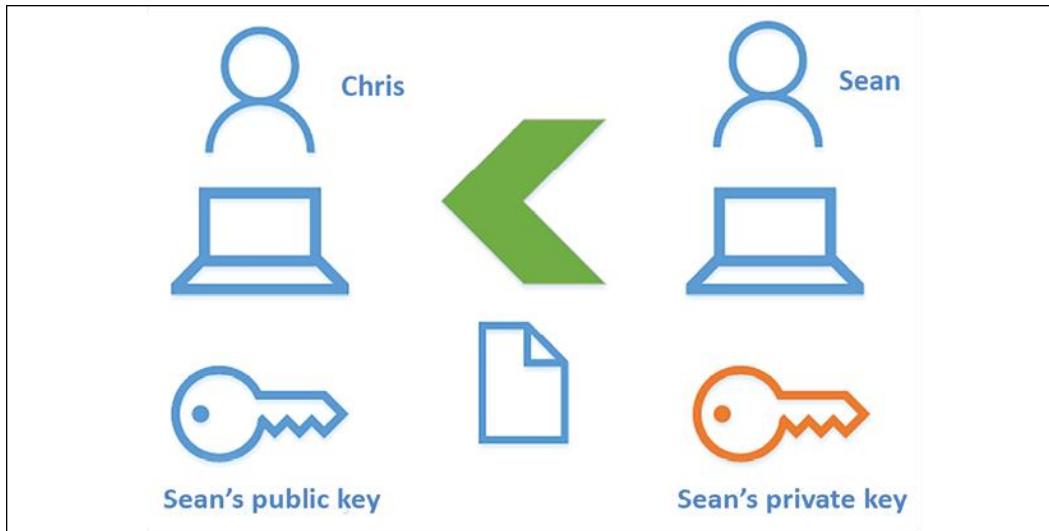


Figure 13.3: Digital signature example

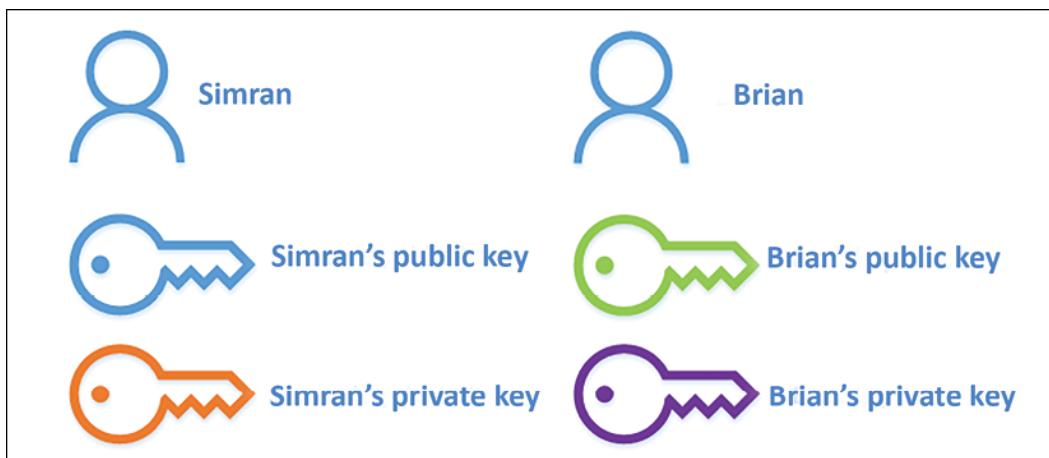


Figure 13.4: An example of key usage

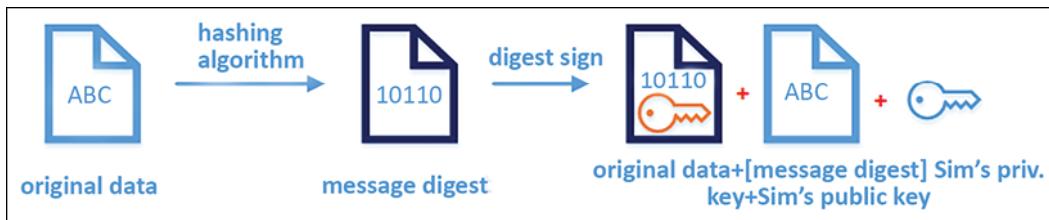


Figure 13.5: Data signing

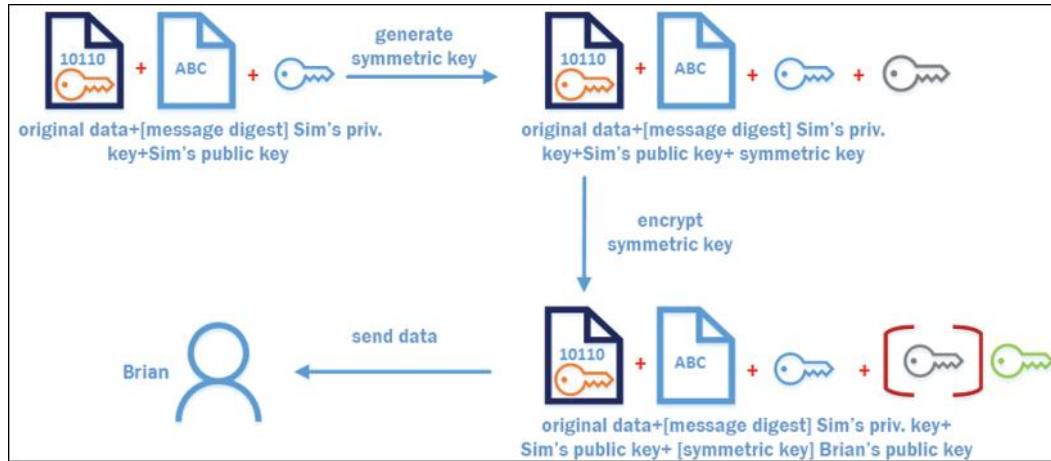


Figure 13.6: Data encryption

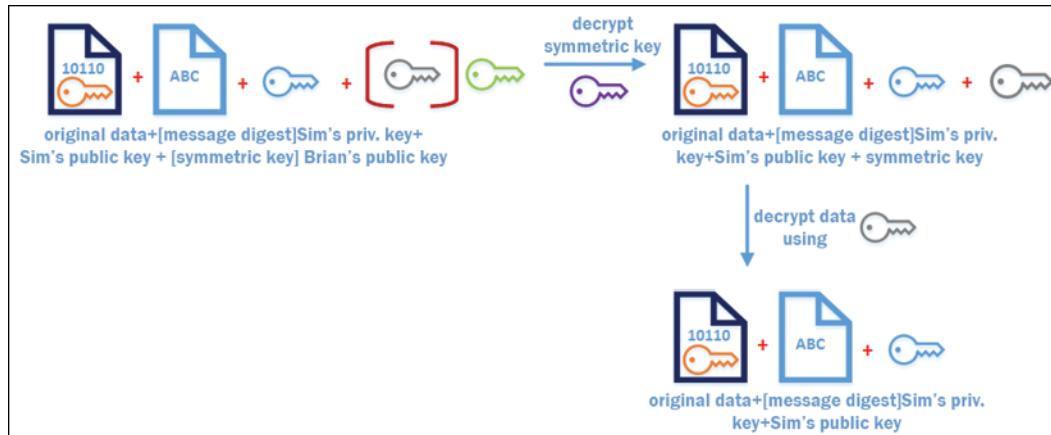


Figure 13.7: Data decryption

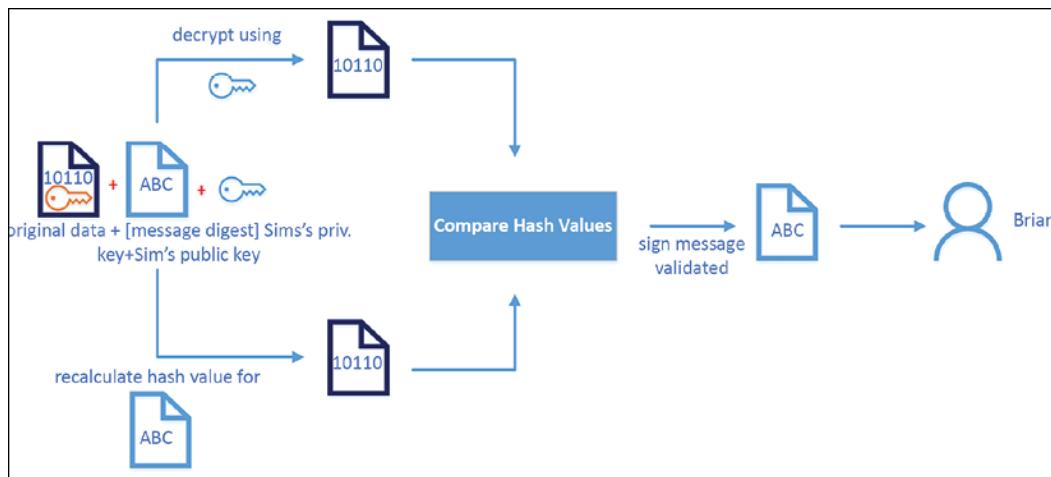


Figure 13.8: Verifying a signature

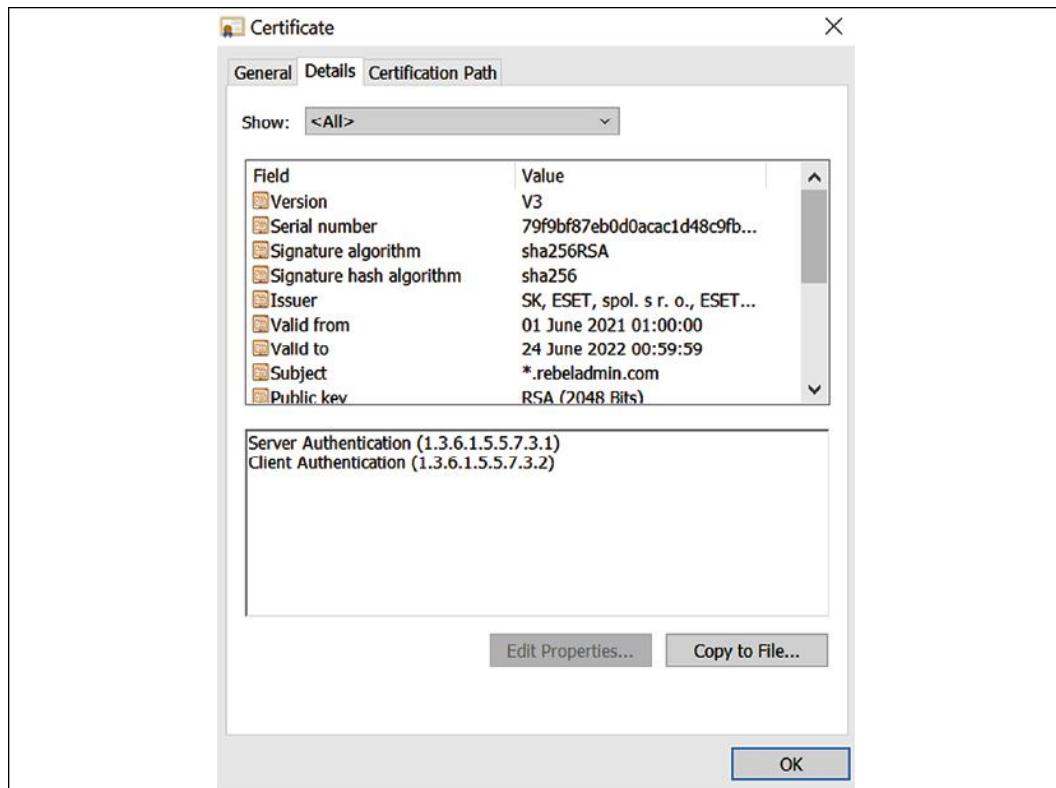


Figure 13.9: Sample certificate

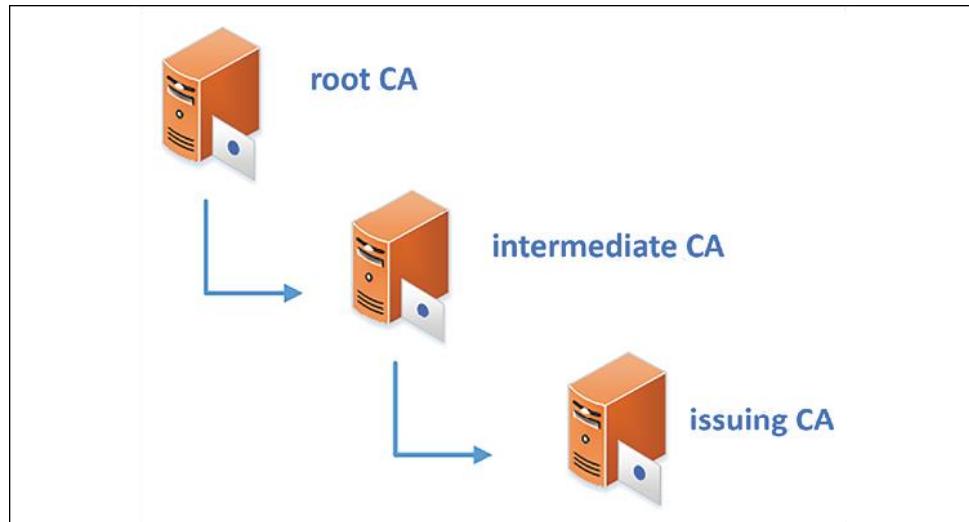


Figure 13.10: CA hierarchy



Figure 13.11: Single-tier model

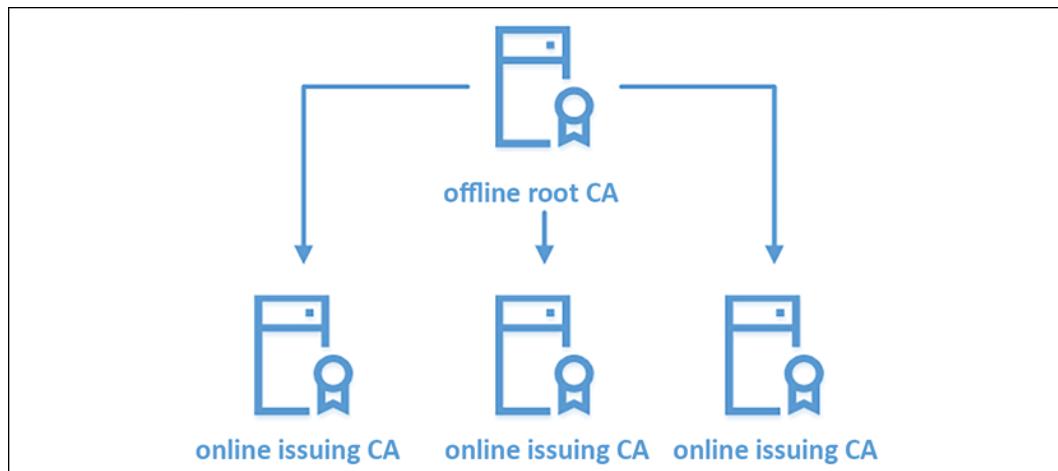


Figure 13.12: Two-tier model

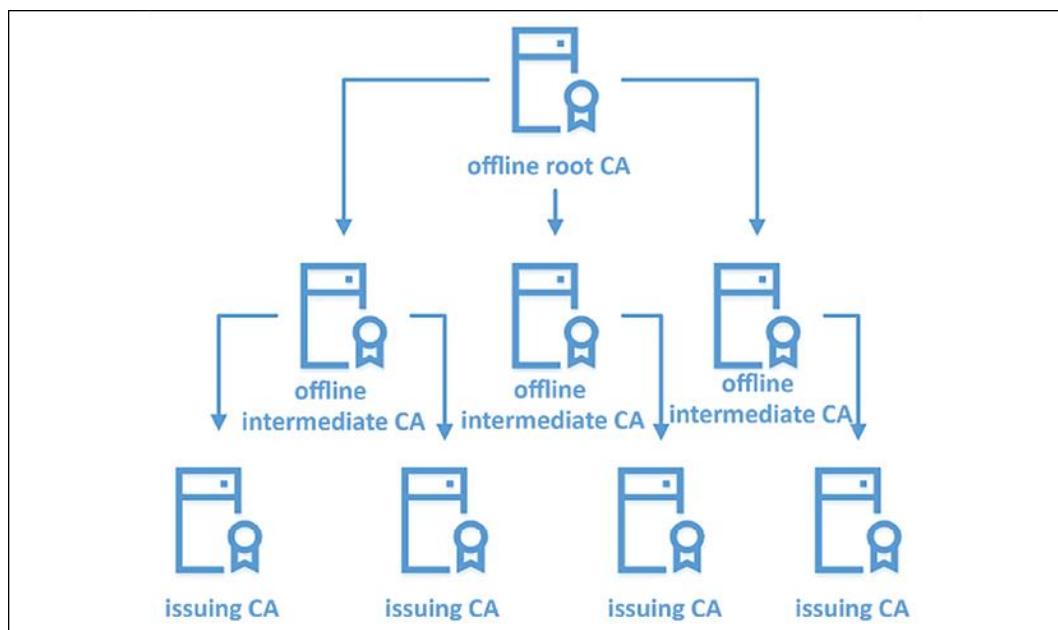


Figure 13.13: Three-tier model

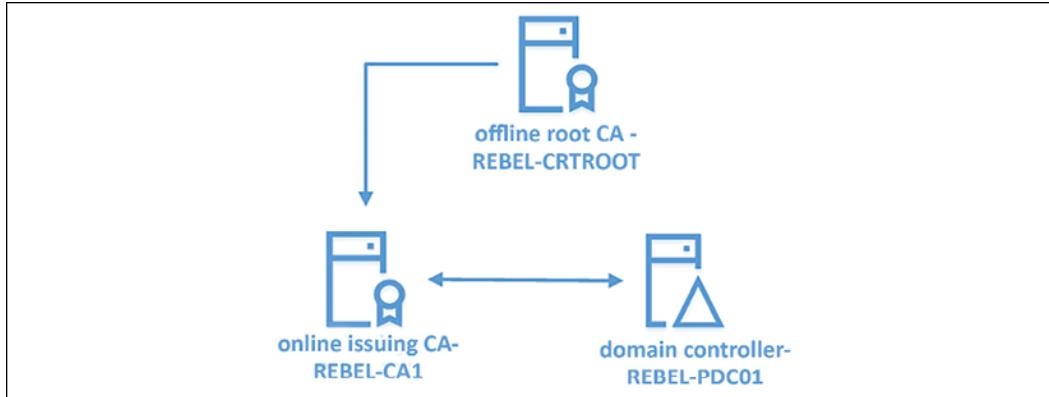


Figure 13.14: Planned PKI setup

```

PS C:\Users\dfrancis> Install-ADcsCertificationAuthority -CACommonName "REBELAdmin Root CA" -CAType StandaloneRootCA -CryptoProviderName "RSA#Microsoft Software Key Storage Provider" -HashAlgorithmName SHA256 -KeyLength 2048 -ValidityPeriod 20 Years -ValidityPeriodUnits 20

Confirm
Are you sure you want to perform this action?
Performing the operation "Install-ADcsCertificationAuthority" on target "REBEL-CRTROOT".
[Y] Yes [A] Yes to All [N] No [L] No to All [?] Help (default is "Y"): A

RunspaceId : cc73471e-e2d8-4756-a194-d50383e01d84
ErrorId     : 0
ErrorString :

```

Figure 13.15: Configuring an AD CS role

```

PS C:\Windows\System32\CertSrv\CertEnroll> dir

Directory: C:\Windows\System32\CertSrv\CertEnroll

Mode                LastWriteTime        Length Name
----                -----          -----
-a---       5/28/2021 11:28 PM           694 REBELAdmin Root CA.crl
-a---       5/28/2021 11:13 PM         793 REBEL-CRTROOT_REBELAdmin Root CA.crt

PS C:\Windows\System32\CertSrv\CertEnroll> _

```

Figure 13.18: Root certificate and CRL

```

Administrator: C:\Program Files\PowerShell\7\pwsh.exe
PowerShell 7.1.4
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Windows\System32> cd c:/
PS C:\> certutil -f -dspublish "REBEL-CRTROOT_ROOT CA.crt" RootCA
ldap:///CN=REBELAdmin Root CA,CN=Certification Authorities,CN=Public Key Services,CN=Services,CN=Configuration,DC=rebeladmin,DC=com?cACertificate

Certificate added to DS store.

ldap:///CN=REBELAdmin Root CA,CN=AIA,CN=Public Key Services,CN=Services,CN=Configuration,DC=rebeladmin,DC=com?cACertificate

Certificate added to DS store.

CertUtil: -dsPublish command completed successfully.
PS C:\>

```

Figure 13.19: Publishing the root CA data

```

Administrator: C:\Program Files\PowerShell\7\pwsh.exe
Type 'help' to get help.

PS C:\Windows\System32> Add-WindowsFeature ADCS-Cert-Authority -IncludeManagementTools
Success Restart Needed Exit Code      Feature Result
----- ----- ----- -----
True    No          Success       {Active Directory Certificate Services, Cert...}

PS C:\Windows\System32> Install-ADcsCertificationAuthority -CACommonName "REBELAdmin IssuingCA" -CAType EnterpriseSubordinateCA -CryptoProviderName "RSA#Microsoft Software Key Storage Provider" -HashAlgorithmName SHA256 -KeyLength 2048

Confirm
Are you sure you want to perform this action?
Performing the operation "Install-ADcsCertificationAuthority" on target "DC22".
[Y] Yes [A] Yes to All [N] No [L] No to All [?] Help (default is "Y"): A
WARNING: The Active Directory Certificate Services installation is incomplete. To complete the installation, use the request file "C:\DC22.rebeladmin.com_REBELAdmin IssuingCA.req" to obtain a certificate from the parent CA. Then, use the Certification Authority snap-in to install the certificate. To complete this procedure, right-click the node with the name of the CA, and then click Install CA Certificate. The operation completed successfully. 0x0 (WIN32: 0)

RunspaceId : 04254b4e-bd45-43a6-ae16-3fd28c7eae5b
ErrorId   : 398
ErrorMessage : The Active Directory Certificate Services installation is incomplete. To complete the installation, use the request file "C:\DC22.rebeladmin.com_REBELAdmin IssuingCA.req" to obtain a certificate from the parent CA. Then, use the Certification Authority snap-in to install the certificate. To complete this procedure, right-click the node with the name of the CA, and then click Install CA Certificate. The operation completed successfully. 0x0 (WIN32: 0)

```

Figure 13.20: Issuing CA role configuration

pkiView - [Enterprise PKI\REBELAdmin Root CA (V0.0)]

Name	Status	Expiration Date
REBELAdmin IssuingCA (V0.0)	OK	5/28/2041 11:12 PM
CA Certificate	OK	5/28/2041 11:12 PM
AIA Location #1	OK	5/28/2041 11:12 PM
AIA Location #2	OK	5/28/2041 11:12 PM
CDP Location #1	OK	8/28/2021 11:38 AM
CDP Location #2	OK	8/28/2021 11:38 AM

Figure 13.21: PKIView

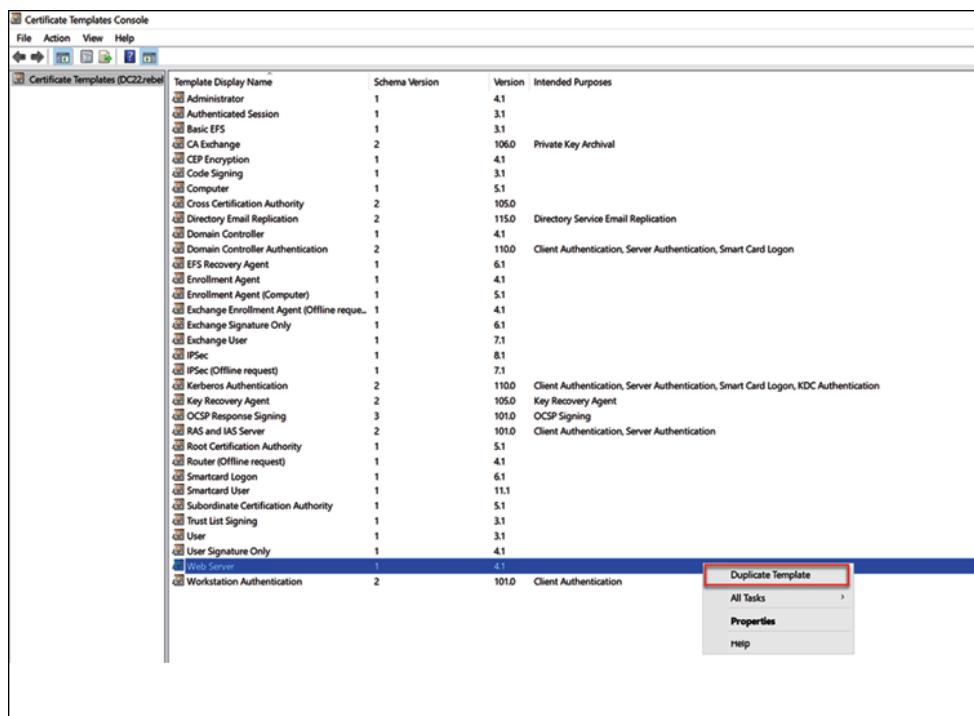


Figure 13.22: Duplicate certificate template

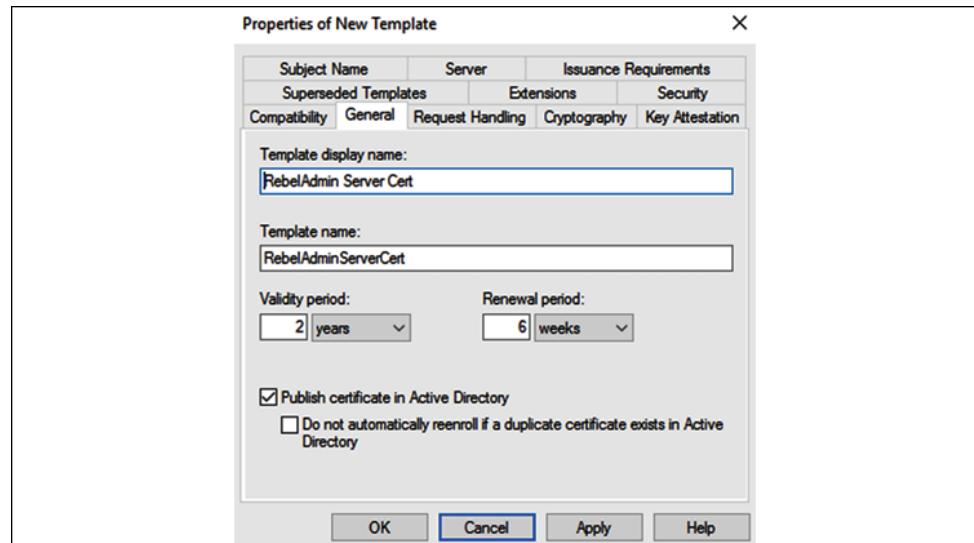


Figure 13.23: Properties of the new template

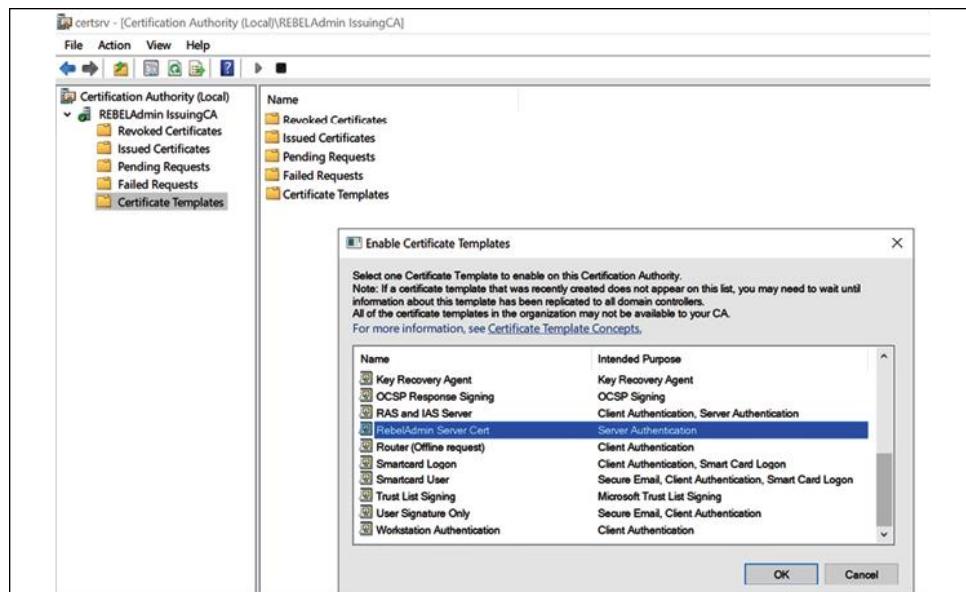


Figure 13.24: Publishing a new template



Figure 13.25: Requesting a new template

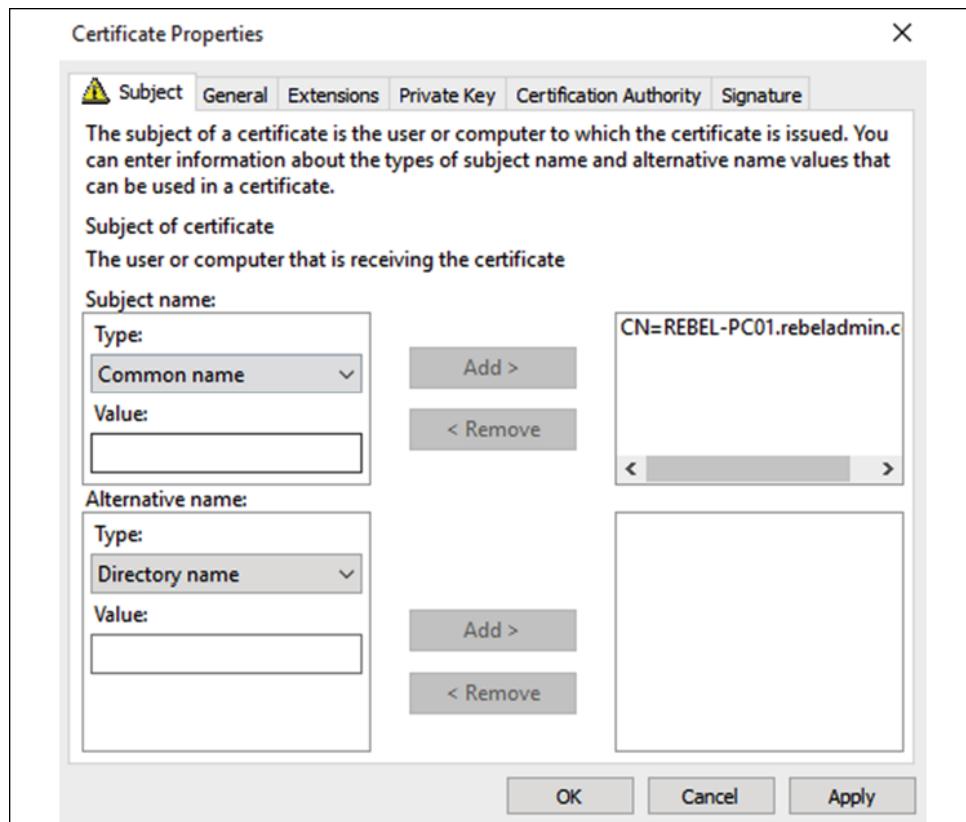


Figure 13.26: Additional information for the certificate request

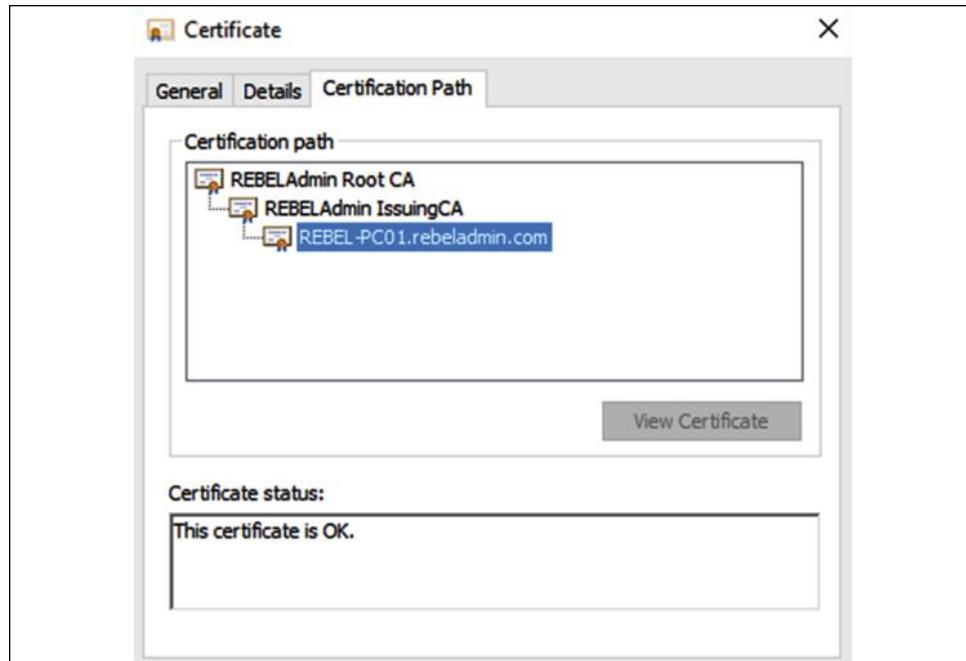


Figure 13.27: Certification path

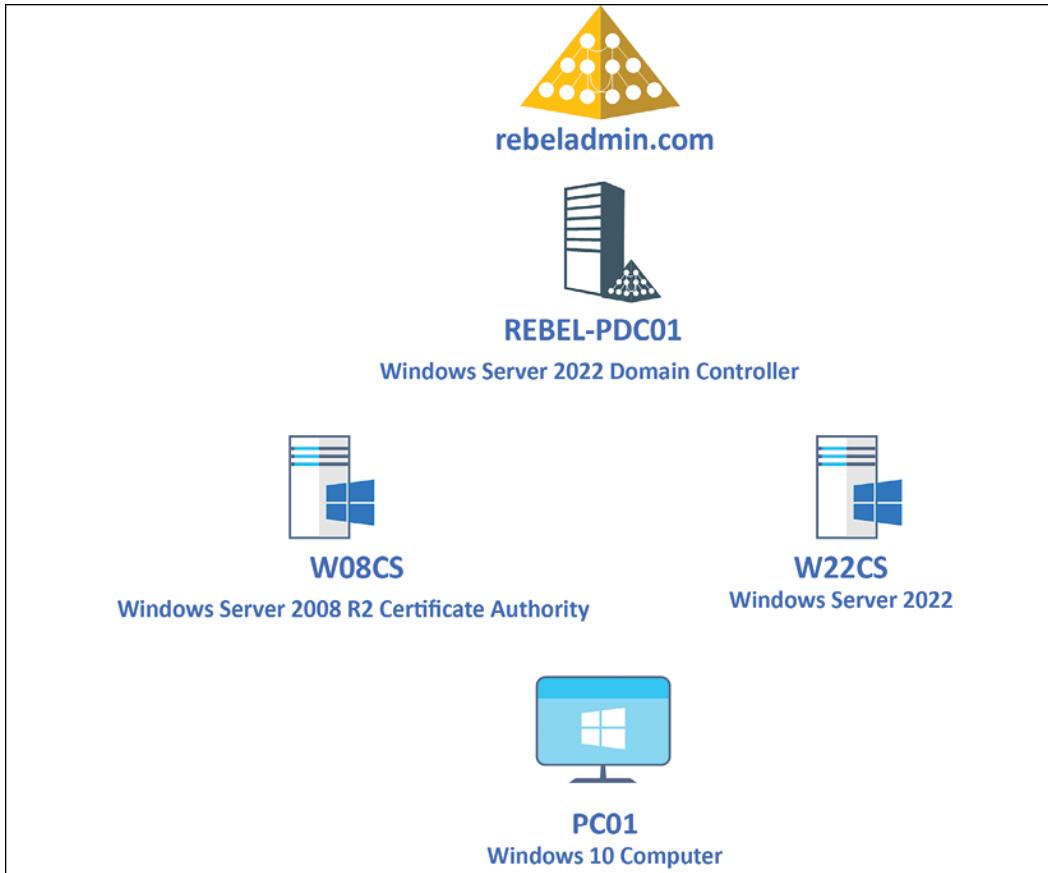


Figure 13.28: Demo environment

```
Administrator: Windows PowerShell
Windows PowerShell
Copyright © 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\dfrancis.REBELADMIN> certutil -backup C:\CABackup
Enter new password:
Confirm new password:
Backed up keys and certificates for W08CS.rebeladmin.com\REBELADMIN CA to C:\CABackup\REBELADMIN CA.p12.
Full database backup for W08CS.rebeladmin.com\REBELADMIN CA.
Backing up Database files: 100%
Backing up Log files: 100%
Truncating Logs: 100%
Backed up database to C:\CABackup.
Database logs successfully truncated.
CertUtil: -backup command completed successfully.
PS C:\Users\dfrancis.REBELADMIN> _
```

Figure 13.29: Backup CA configuration

Name	Date modified	Type	Size
Database	9/13/2021 9:53 PM	File folder	
REBELAdmin IssuingCA	9/13/2021 9:53 PM	Personal Information	4 KB

Figure 13.30: CA backup content

```
PS C:\Users\dfrancis.REBELADMIN> Install-AdcsCertificationAuthority -CAType EnterpriseRootCa -CertFile "C:\CABackup\REBELADMIN CA.p12" -CertFilePassword (read-host "Cert Password" -assecurestring)
Cert Password: *****

Confirm
Are you sure you want to perform this action?
Performing the operation "Install-AdcsCertificationAuthority" on target "W19CS".
[Y] Yes [A] Yes to All [N] No [L] No to All [?] Help (default is "Y"): A

RunspaceId : b448b170-43cd-4942-a5cc-e904851ff155
ErrorId : 0
ErrorException :
```

Figure 13.31: Configuring an AD CS role with the existing CA certificate

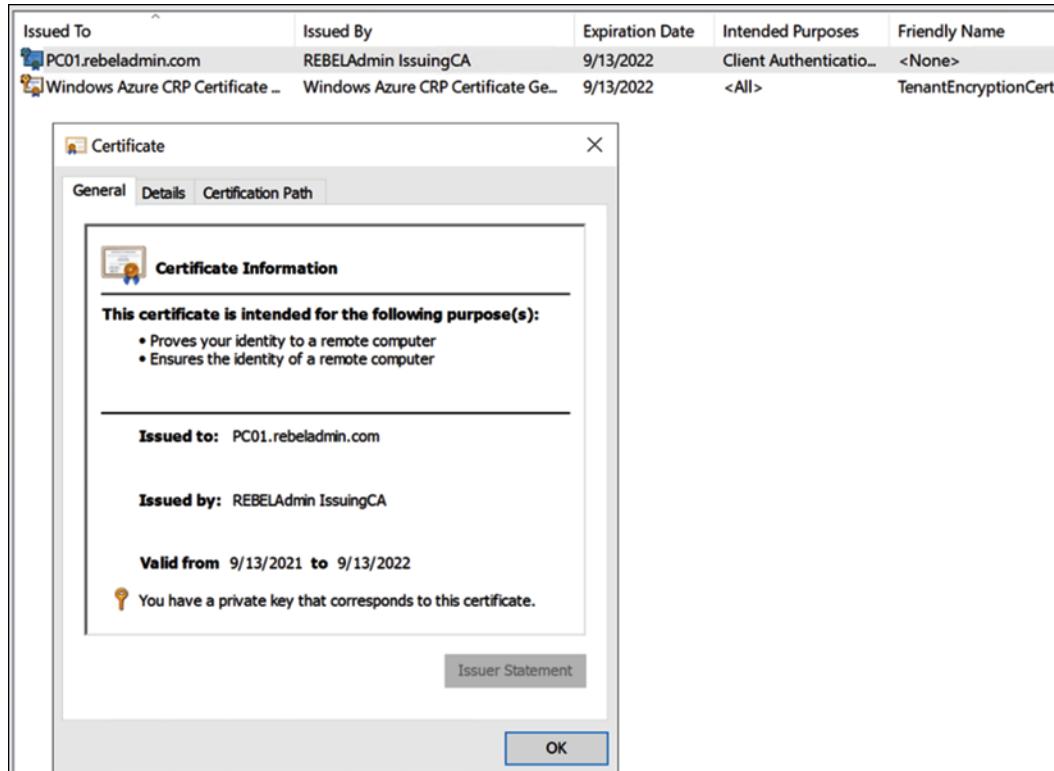


Figure 13.32: Existing certificate

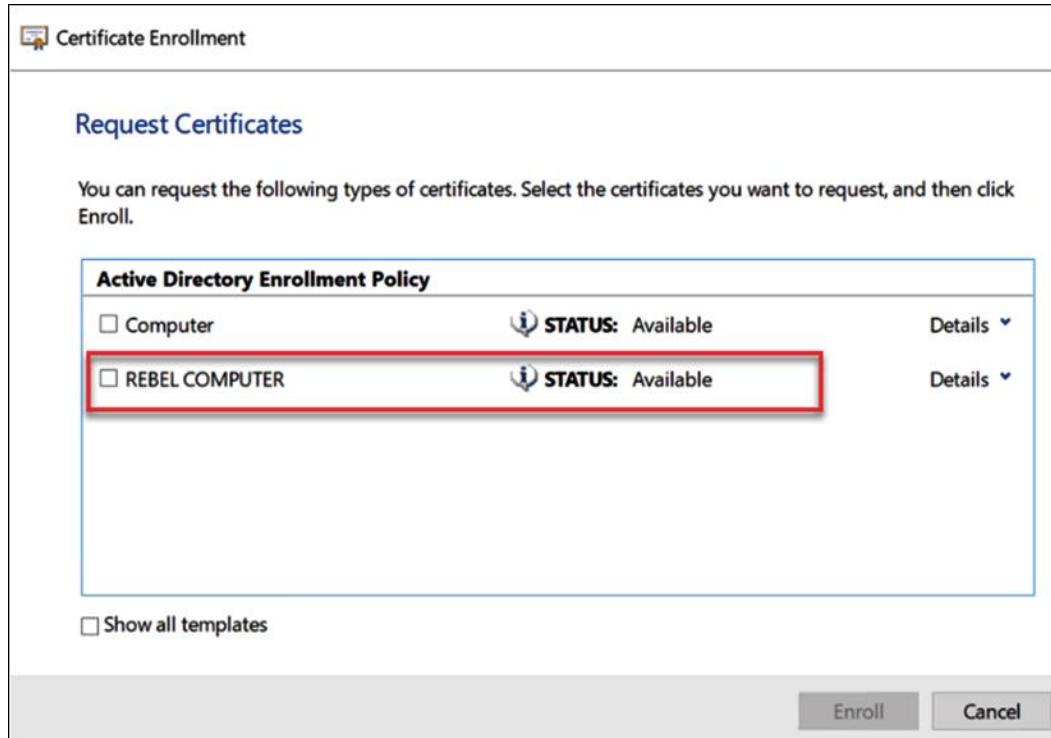


Figure 13.33: Requesting a new certificate

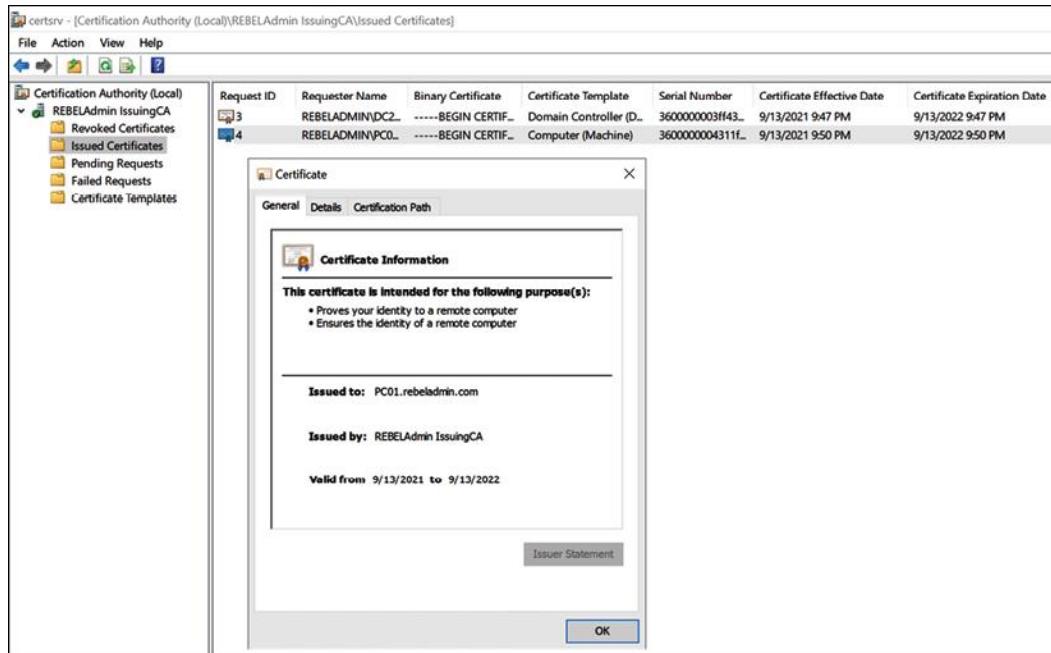


Figure 13.34: New certificate from the Windows Server 2022 CA

```

Administrator: PowerShell 7 (x64)
PowerShell 7.1.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Users\dfrancis.REBELADMIN> Backup-CARoleService C:\CABackup
PS C:\Users\dfrancis.REBELADMIN>

```

Figure 13.35: Backup CA configuration

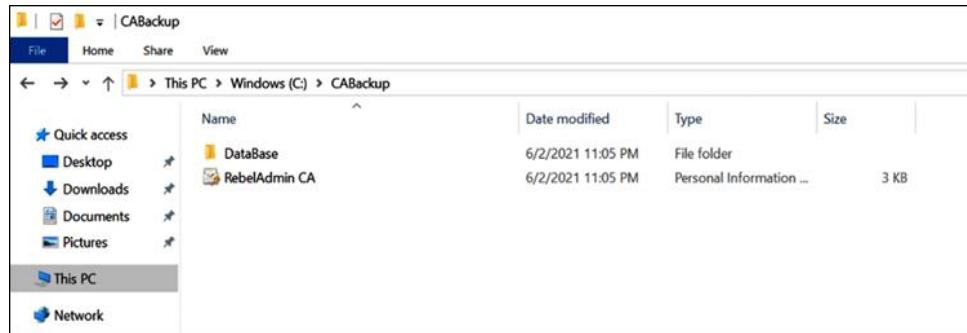


Figure 13.36: File in the CABackup location

Tables

Feature	Standalone CA	Enterprise CA
AD DS dependency	Does not depend on AD DS; can be installed on a member server or standalone server in a workgroup	Can only be installed on a member server
Operate offline	Can stay offline	Cannot be offline
Customized certificate templates	Only supports standard templates	Supported
Supported enrollment methods	Manual or web enrollment	Auto, manual, or web enrollment
Certificate approval process	Manual	Manual or automatic based on the policy
User input for certificate fields	Manual	Retrieved from AD DS
Certificate issuing and managing using AD DS	N/A	Supported

Table 13.1 – the types of CA

Option	Details
0	No changes.
1	Publish a CA certificate to a given location.
2	Attach the AIA extensions of issued certificates.
32	Attach the Online Certificate Status Protocol (OCSP) extensions.

Table 13.2 – AIA locations

Host Name	Operating System	Role
REBEL-PDC01	Windows Server 2022	Primary Domain Controller in rebeladmin.com Active Directory Domain.
W08CS	Windows Server 2008 R2	Existing CA.
W22CS	Windows Server 2022	After AD CS configuration is migrated, this server will become the CA in the rebeladmin.com domain.
PC01	Windows 10	Test PC.

Table 13.3 – Role of server and PC

Commands

Command 13.1

The first task is to install the AD CS role service. This can be done using the following command:

```
Add-WindowsFeature ADCS-Cert-Authority -IncludeManagementTools
```

Command 13.2

Once the role service is installed, the next step is to configure the role and get the CA up and running:

```
Install-ADcsCertificationAuthority -CACommonName "REBELAdmin Root CA"
-CAType StandaloneRootCA -CryptoProviderName "RSA#Microsoft Software
Key Storage Provider" -HashAlgorithmName SHA256 -KeyLength 2048 -
ValidityPeriod Years -ValidityPeriodUnits 20
```

Command 13.3

```
certutil.exe -setreg ca\DSConfigDN
CN=Configuration,DC=rebeladmin,DC=com
```

The preceding command needs to run using Command Prompt.

Command 13.4

The AIA location can be set using the certutil command.

```
certutil -setreg CA\CACertPublicationURLs  
"1:C:\Windows\system32\CertSrv\CertEnroll\%1_%3%4.crt\n2:ldap:///CN=%7  
,CN=AIA,CN=Public Key  
Services,CN=Services,%6%11\n2:http://crt.rebeladmin.com/CertEnroll/%1_  
%3%4.crt"
```

Command 13.5

Setting CA time limits - For this demo, I will set the certificate validity period to 10 years:

```
certutil -setreg ca\ValidityPeriod "Years"  
certutil -setreg ca\ValidityPeriodUnits 10
```

Command 13.6

Now we have all the settings submitted. To apply the changes, run the following command:

```
restart-service certsvc
```

Command 13.7

The next step is to create a new CRL, which can be generated using the following command:

```
certutil -crl
```

Command 13.8

Then, log in to the domain controller as Domain Admin or Enterprise Admin and run the following command:

```
certutil -f -dspublish "REBEL-CRTROOT_ROOTCA.crt" RootCA
```

Command 13.9

This also needs to be published to Active Directory so that everyone in the domain is aware of it. To do that, copy the file from the root CA to the domain controller and run the following command:

```
certutil -f -dspublish "REBELAdmin Root CA.crl"
```

Command 13.10

The file needs to be copied from the issuing CA to the root CA, and then you need to execute the following command:

```
certreq -submit "REBEL-CA1.rebeladmin.com_REBELAdmin_IssuingCA.req"
```

Command 13.11

Once it has been issued, it needs to be exported and imported into the issuing CA:

```
certreq -retrieve 2 "C:\REBEL-  
CA1.rebeladmin.com_REBELAdmin_IssuingCA.crt"
```

The preceding command will export the certificate. The number 2 is the request ID in the CA Microsoft Management Console (MMC).

Command 13.12

Once the export is complete, move the file to the issuing CA, and from there, run the cert util install cert command.

```
Certutil -installcert "C:\REBEL-  
CA1.rebeladmin.com_REBELAdmin_IssuingCA.crt" start-service certsvc
```

Command 13.13

```
certutil -setreg CA\CACertPublicationURLs "1:  
C:\Windows\system32\CertSrv\CertEnroll\%1_%3%4.crt\n2:http://crt.rebel  
admin.com/CertEnroll/%1_%3%4.crt\n3:ldap://CN=%7,CN=AIA,CN=Public Key  
Services,CN=Services,%6%11"
```

Command 13.14

```
certutil -setreg CA\CRLPeriodUnits 7  
certutil -setreg CA\CRLPeriod "Days"  
certutil -setreg CA\CRLOverlapPeriodUnits 3  
certutil -setreg CA\CRLOverlapPeriod "Days"  
certutil -setreg CA\CRLDeltaPeriodUnits 0  
certutil -setreg ca\ValidityPeriodUnits 3  
certutil -setreg ca\ValidityPeriod "Years"
```

Command 13.15

Once all this is done, in order to complete the configuration, restart the certificate service using the following command:

```
restart-service certsvc
```

Command 13.16

Last but not least, run the following command to generate the CRLs:

```
certutil -crl
```

Command 13.17

We also need to export the CA configuration settings saved under the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration registry key. To export the key, run the following PowerShell command:

```
reg.exe export HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration C:\CABackup\careg.reg
```

The preceding command will back up the registry key to the C:\CABackup folder and save it as careg.reg.

Command 13.18

Run the following command to install the AD CS role:

```
Add-WindowsFeature ADCS-Cert-Authority -IncludeManagementTools
```

Command 13.19

Run the following command to configure AD CS role with the existing CA certificate:

```
Install-AdcsCertificationAuthority -CAType EnterpriseRootCa -CertFile "C:\CABackup\REBELADMIN CA.p12" -CertFilePath (read-host "Cert Password" -assecurestring) :
```

The preceding command configures the AD CS role with the existing CA certificate, which is saved as C:\CABackup\REBELADMIN CA.p12 from the previous CA backup.

CDP locations

CDPs define the location from where the CRL can be retrieved. This is a web-based location and should be accessible via HTTP. This list will be used by the certificate validator to verify the given certificate against the revocation list.

Before we do this, we need to prepare the web server. It should be a domain member, similar to the issuing CA.

In my demonstration, I am going to use the same issuing CA as the CDP location.

The web server can be installed using the following command:

```
Install-WindowsFeature Web-WebServer -IncludeManagementTools
```

Next, create a folder and create a share so that it can be used as the virtual directory:

```
mkdir C:\CertEnroll  
New-smbshare -name CertEnroll C:\CertEnroll -FullAccess  
SYSTEM,"rebeladmin\Domain Admins" -ChangeAccess "rebeladmin\Cert  
Publishers"
```

As part of the exercise, I am setting the share permissions to rebeladmin\Domain Admins (full access) and rebeladmin\Cert Publishers (change access).

After the folder has been created with the relevant permissions, we also need to copy the root CA certificate and the CRL file to it:

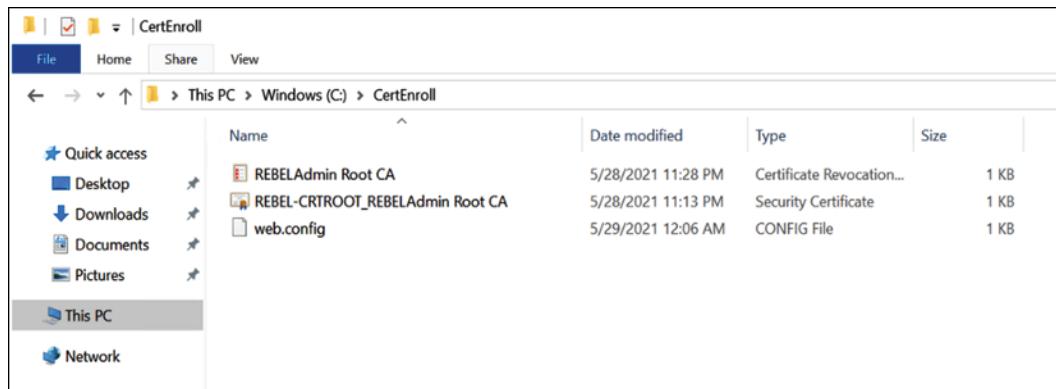


Figure 13.16: Configuring an AD CS role

After that, load the **Internet Information Services (IIS)** manager and add a virtual directory, CertEnroll, with the aforementioned path:

CRL time limits

The CRL also has some associated time limits:

```
Certutil -setreg CA\CRLPeriodUnits 13  
Certutil -setreg CA\CRLPeriod "Weeks"  
Certutil -setreg CA\CRLDeltaPeriodUnits 0  
Certutil -setreg CA\CRLOverlapPeriodUnits 6  
Certutil -setreg CA\CRLOverlapPeriod "Hours"
```

In the preceding commands, the following is true:

- **CRLPeriodUnits**: This specifies the number of days, weeks, months, or years for which the CRL will be valid.
- **CRLPeriod**: This specifies whether the CRL validity period is measured by days, weeks, months, or years.
- **CRLDeltaPeriodUnits**: This specifies the number of days, weeks, months, or years that the delta CRL is valid for. Offline CAs should disable this.
- **CRLOverlapPeriodUnits**: This specifies the number of days, weeks, months, or years that the CRL can overlap.
- **CRLOverlapPeriod**: This specifies whether the CRL overlapping validity period is measured by days, weeks, months, or years.

Now we have all the settings submitted. To apply the changes, run the following command:

```
restart-service certsvc
```

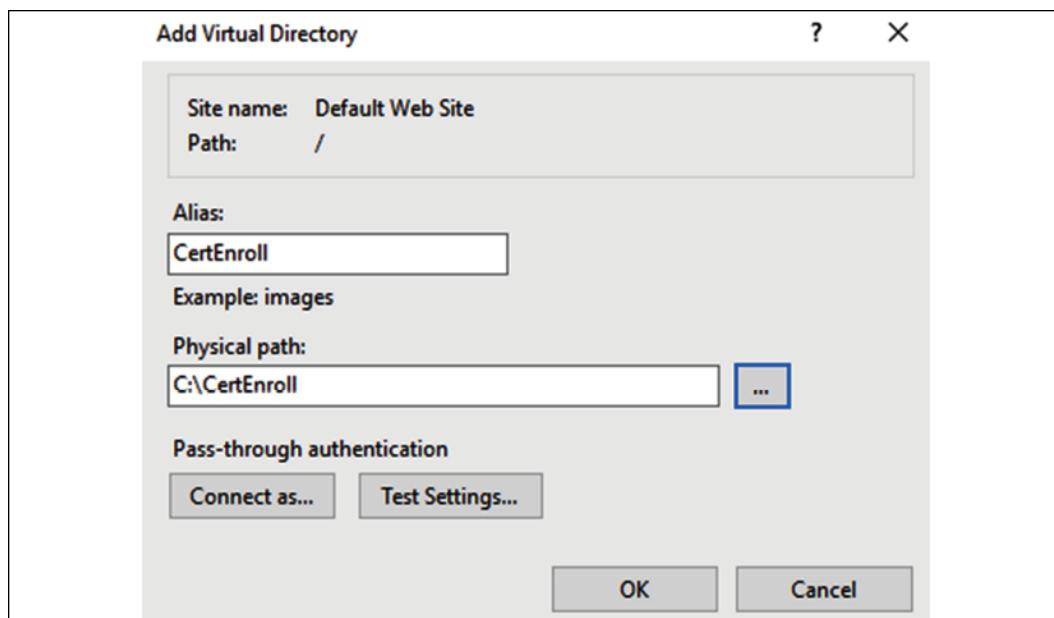


Figure 13.17: Setting up the CertEnroll virtual directory

Last but not least, we need to create a DNS record for the service URL. In this demo, I am using **crt.rebeladmin.com** as the domain name. This will allow us to access the new distribution point using **http://crt.rebeladmin.com/CertEnroll**.

Now everything is ready, and we can publish the CDP settings using the following command:

```
certutil -setreg CA\CRLPublicationURLs
"1:C:\Windows\system32\CertSrv\CertEnroll\%3%8%9.crl
\n10:ldap:///CN=%7%8,CN=%2,CN=CDP,CN=Public Key
```

```
Services,CN=Services,%6%10\n2:http://crt.rebeladmin.com/CertEnroll/%3%8%9.crl"
```

The preceding command needs to run on the root CA server.

The single numbers in the command refer to the options, and numbers with % refer to the variables:

Option	Details
0	No changes.
1	Publish the CRL to the given location.
2	Attach the CDP extensions of issued certificates.
4	Include in the CRL to find the delta CRL locations.
8	Specify whether there is a need to publish all CRL information to AD when publishing manually.
64	Delta CRL location.
128	Include the Issuing Distribution Point (IDP) extension of the issued CRL.

All these settings can be specified using the GUI. To access it, go to **Server Manager | Tools | Certification Authority**, right-click and select **Properties** of the server, and then go to the **Extension** tab.

There, you can add the following variables using the GUI:

Variable	GUI reference	Details
%1	<ServerDNSName>	The DNS name of the CA server
%2	<ServerShortName>	The NetBIOS name of the CA server
%3	<CAName>	The given name for the CA
%4	<CertificateName>	The renewal extension of the CA
%6	<ConfigurationContainer>	DN of the configuration container in AD
%7	<CATruncatedName>	Truncated name of the CA (32 characters)
%8	<CRLNameSuffix>	Inserts a name suffix at the end of the filename before publishing a CRL
%9	<DeltaCRLAllowed>	Replaces CRLNameSuffix with a separate suffix to use the delta CRL
%10	<CDPOBJECTCLASS>	The object class identifier for the CDP
%11	<CAOBJECTCLASS>	The object class identifier for a CA

Once the CDP settings are in place, the next step is to go ahead and set up the AIA locations.

Setting up the issuing CA

Now that we're done with the root CA setup, the next step is to set up the issuing CA. Issuing CAs will be run from a domain member server and will be AD-integrated. To begin the installation, log in to the server as the Domain Admin or Enterprise Admin.

1. The first task will be to install the AD CS role:

```
Add-WindowsFeature ADCS-Cert-Authority -IncludeManagementTools
```

2. I will use the same server for the Web Enrollment Role Service. This can be added using the following command:

```
Add-WindowsFeature ADCS-web-enrollment
```

3. After that, we can configure the role service using the following command:

```
Install-ADcsCertificationAuthority -CACoName "REBELAdmin  
IssuingCA" -CAType EnterpriseSubordinateCA -CryptoProviderName  
"RSA#Microsoft Software Key Storage Provider" -HashAlgorithmName  
SHA256 -KeyLength 2048
```

4. To configure the Web Enrollment Role Service, use the following command:

```
Install-ADCSwebenrollment
```

5. This completes the initial role configuration process of the issuing CA. The next step is to create a certificate for the issuing CA.

Chapter 14

Figures

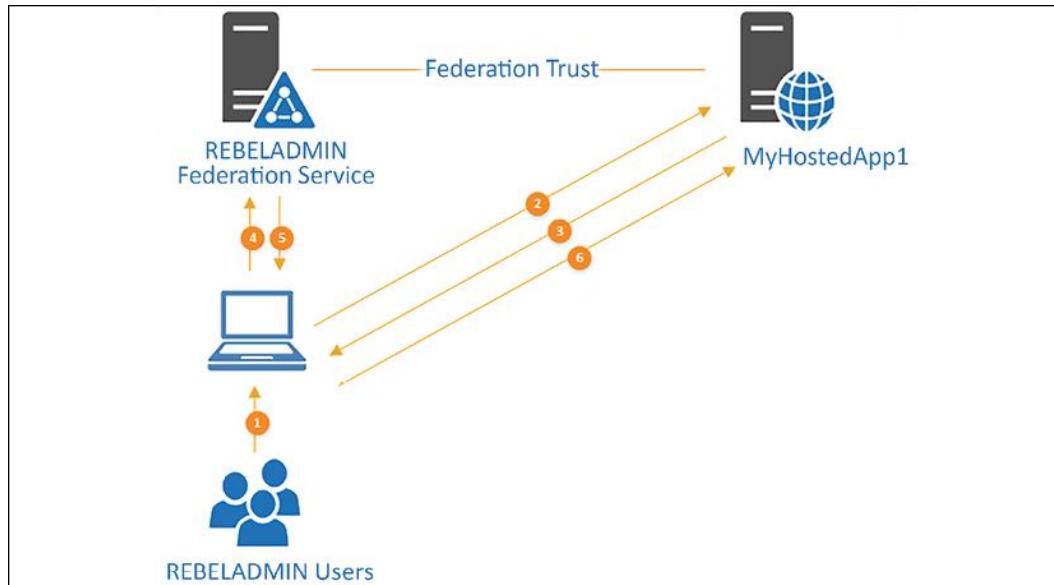


Figure 14.2: Federation trust in action

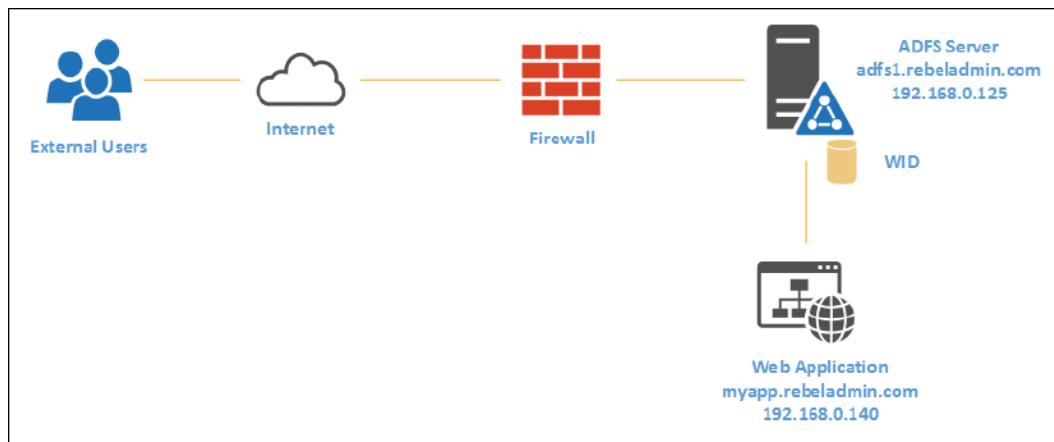


Figure 14.3: Single federation server deployment

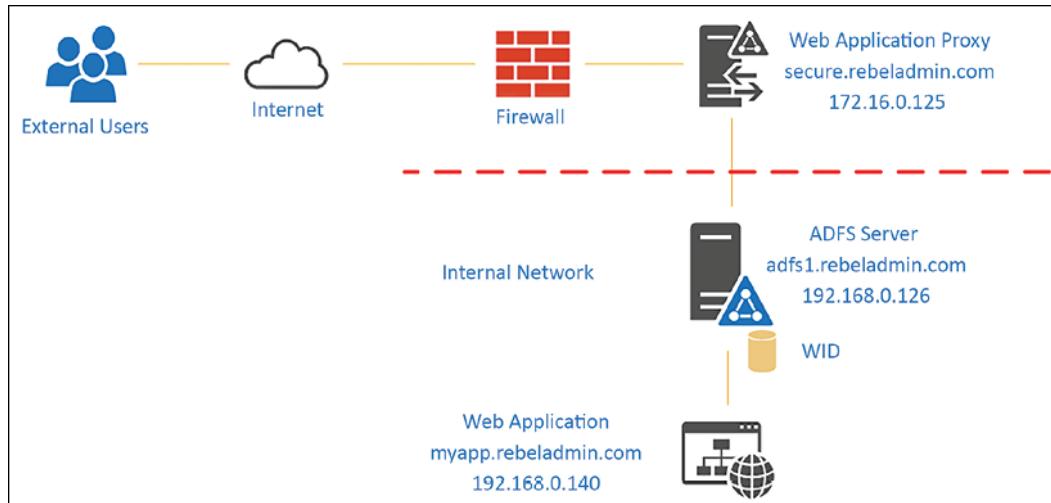


Figure 14.4: Single federation server and WAP deployment

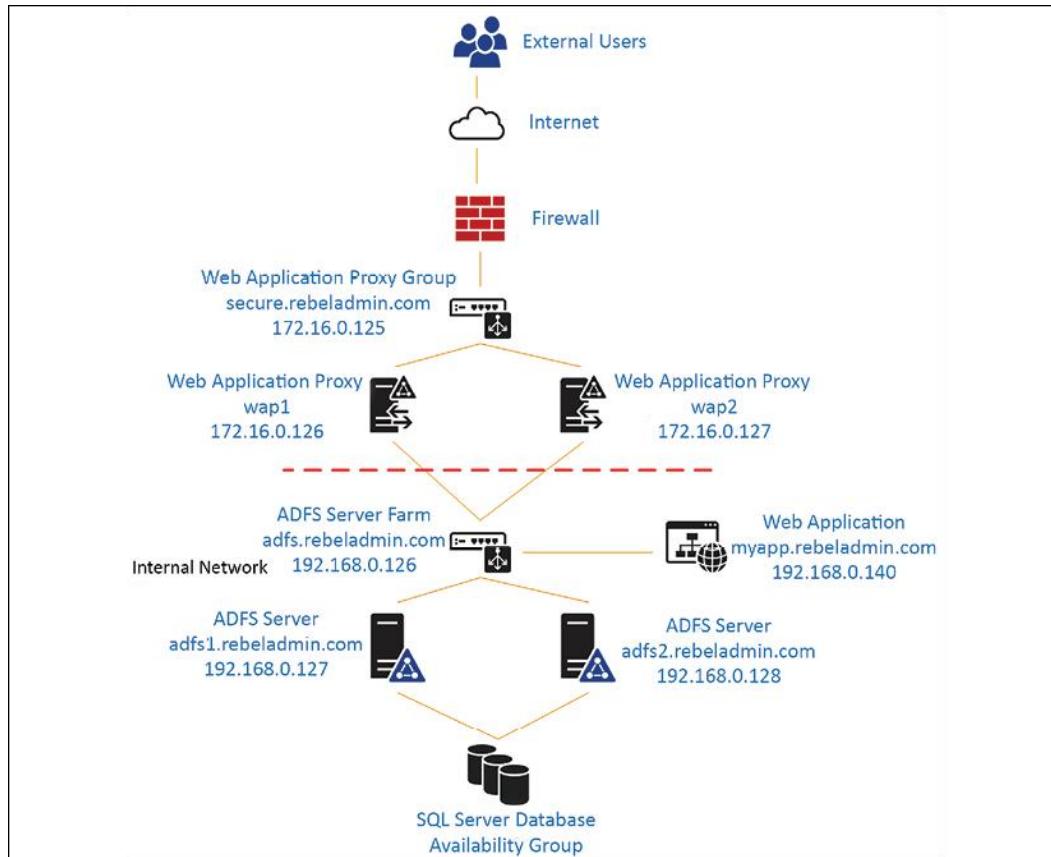


Figure 14.5: A multiple federation server and multiple WAP server deployment

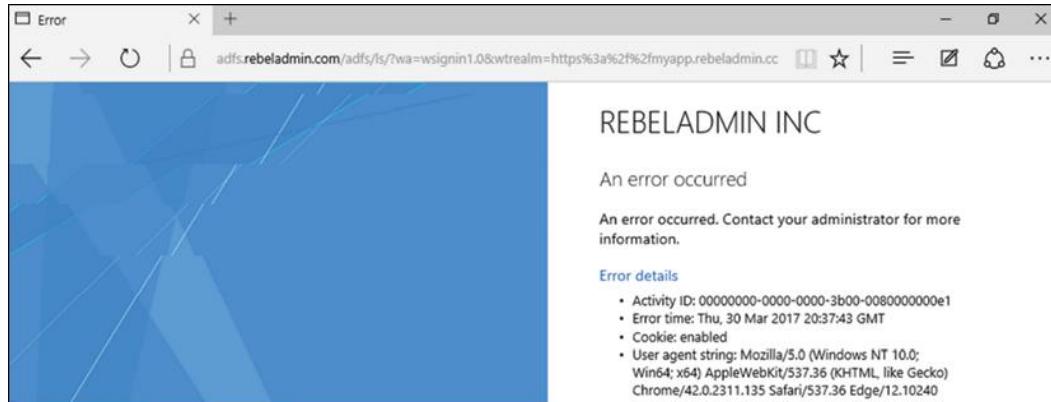


Figure 14.8: AD FS login page

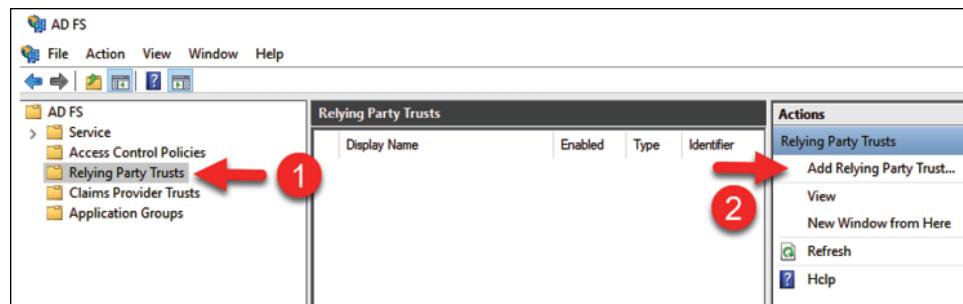


Figure 14.9: Add Relying Party Trust

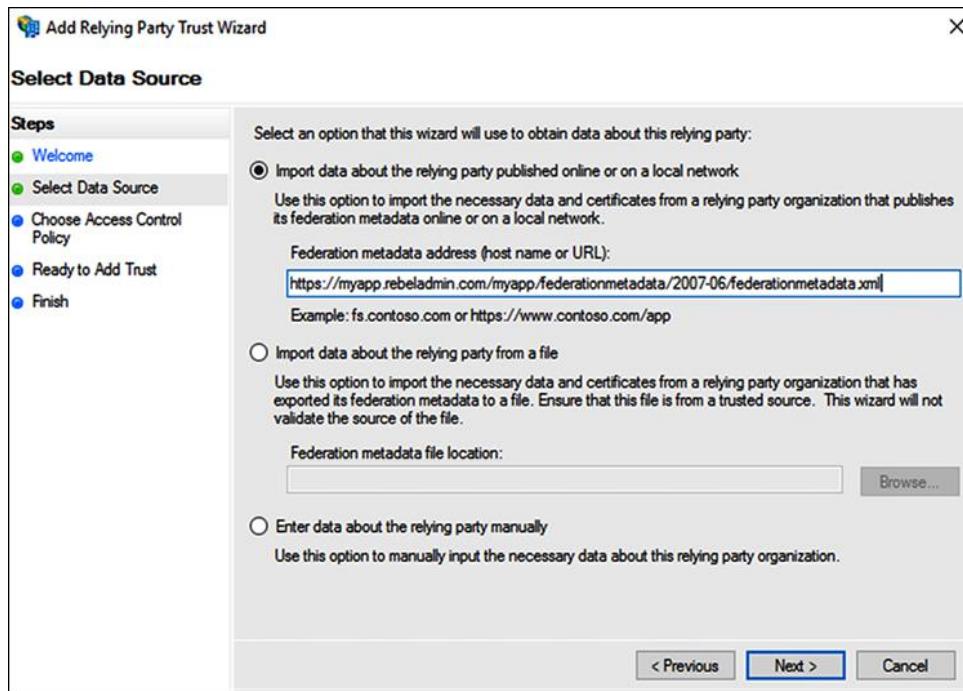


Figure 14.10: Configure the relying party trust

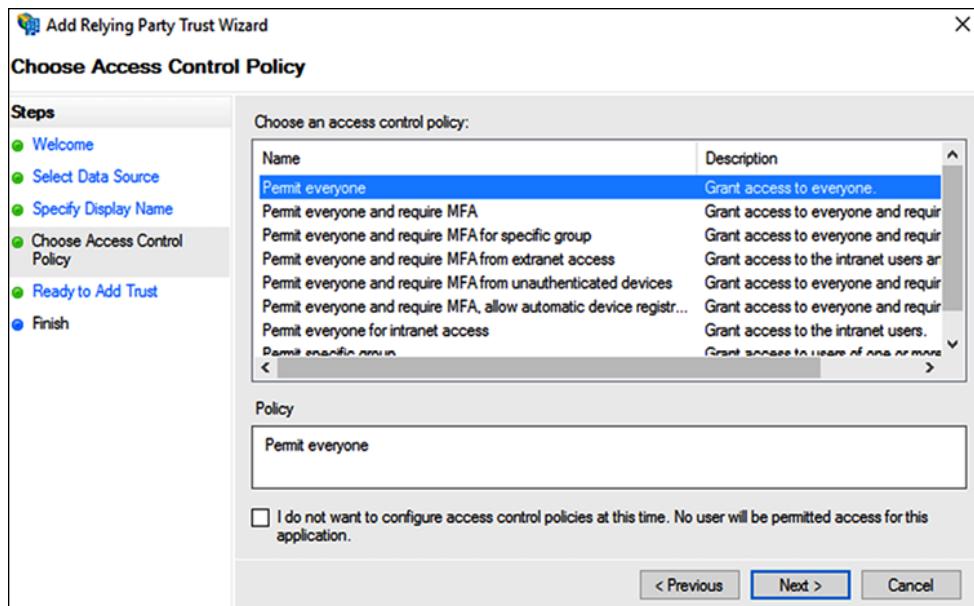


Figure 14.11: Relying party trust access control policies

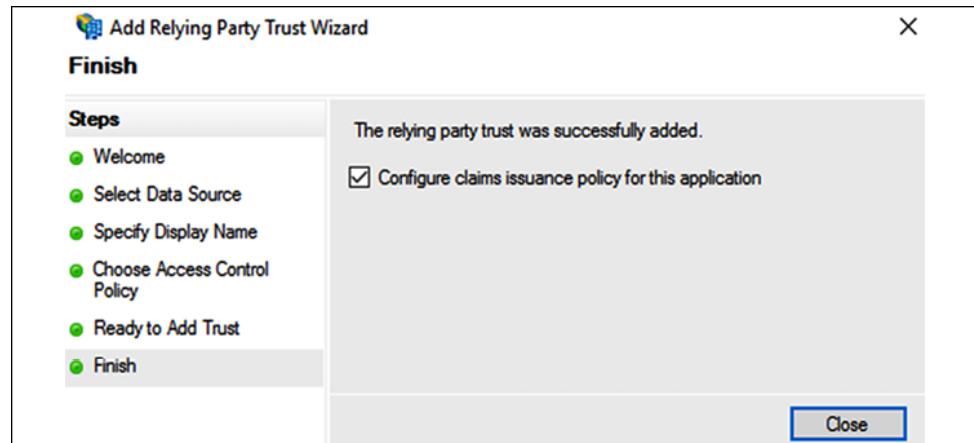


Figure 14.12: Completing the relying party trust wizard

Claim Type	Claim Value	Type	Sub
http://schemas.microsoft.com/ws/2014/01/identity/claims/anchorclaimtype	http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname	string	REBEL/
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/implicitupn	dfrancis@rebeladmin.com	string	REBEL/
http://schemas.microsoft.com/claims/authnmethodsproviders	WindowsAuthentication	string	REBEL/
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn	dfrancis@rebeladmin.com	string	REBEL/
http://schemas.microsoft.com/ws/2008/06/identity/claims/primarygroupsid	S-1-5-21-4041220333-1835452706-552999228-513	string	REBEL/
http://schemas.microsoft.com/ws/2008/06/identity/claims/groupsid	S-1-5-21-4041220333-1835452706-552999228-513	string	REBEL/
http://schemas.microsoft.com/ws/2008/06/identity/claims/groupsid	S-1-1-0	string	REBEL/
http://schemas.microsoft.com/ws/2008/06/identity/claims/groupsid	S-1-5-32-545	string	REBEL/
http://schemas.microsoft.com/ws/2008/06/identity/claims/groupsid	S-1-5-2	string	REBEL/
http://schemas.microsoft.com/ws/2008/06/identity/claims/groupsid	S-1-5-11	string	REBEL/
http://schemas.microsoft.com/ws/2008/06/identity/claims/groupsid	S-1-5-15	string	REBEL/
http://schemas.microsoft.com/ws/2008/06/identity/claims/groupsid	S-1-18-1	string	REBEL/
http://schemas.microsoft.com/ws/2008/06/identity/claims/primarysid	S-1-5-21-4041220333-1835452706-552999228-1186	string	REBEL/
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	REBELADMIN\dfrancis	string	REBEL/
http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname	REBELADMIN\dfrancis	string	REBEL/
http://schemas.microsoft.com/claims/authnmethodreferencedescriptions	http://schemas.microsoft.com/ws/2008/06/identity/authenticationmethod/windows	string	REBEL/
http://schemas.microsoft.com/claims/authnmethodreferencedescriptions	http://schemas.microsoft.com/ws/2008/06/identity/authenticationmethod/kerberos	string	REBEL/
http://schemas.microsoft.com/2012/01/requestcontext/claims/x-ms-client-user-agent	Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko	string	REBEL/

Figure 14.13: Application access

```

Environment      : AzureCloud
Account         : dcadmin@REBELADMIN.onmicrosoft.com
TenantId        : 05c6f80c-61d9-44df-bd2d-4414a983c1d4
SubscriptionId   :
SubscriptionName :
CurrentStorageAccount :

```

Figure 14.14: Azure TenantId

Issued To	Issued By	Expiration Date	Intended Purposes	Friendly Name
05c6f80c-61d9-44df-bd2d-4414a983c1d4	rebeladmin-DC01-CA	7/28/2021	Client Authentication	<None>
DC01.rebeladmin.com	rebeladmin-DC01-CA	7/25/2020	Client Authentication	<None>
rebeladmin-DC01-CA	rebeladmin-DC01-CA	7/26/2024	<All>	<None>

Figure 14.15: New certificate

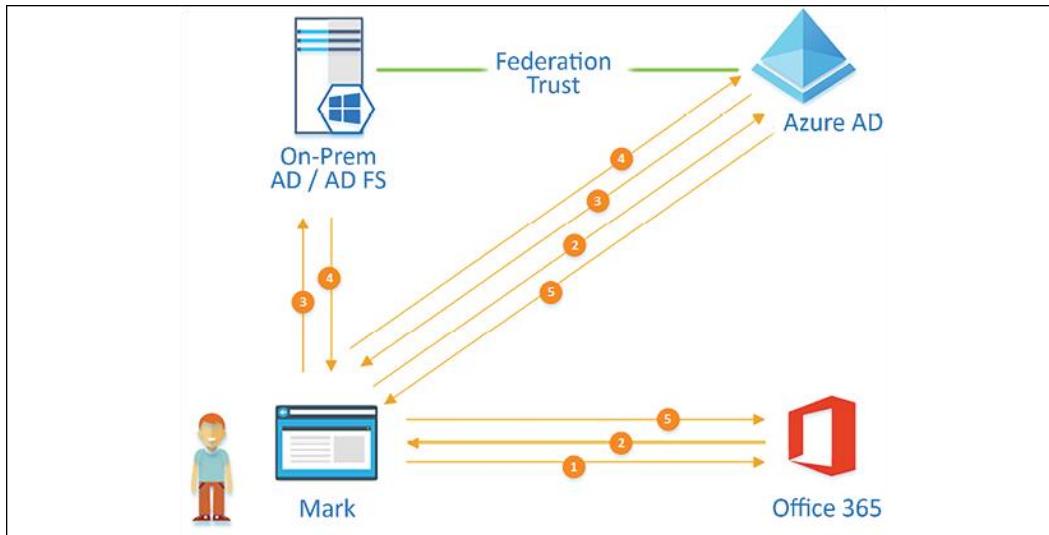


Figure 14.20: How AD FS federation works with Azure AD

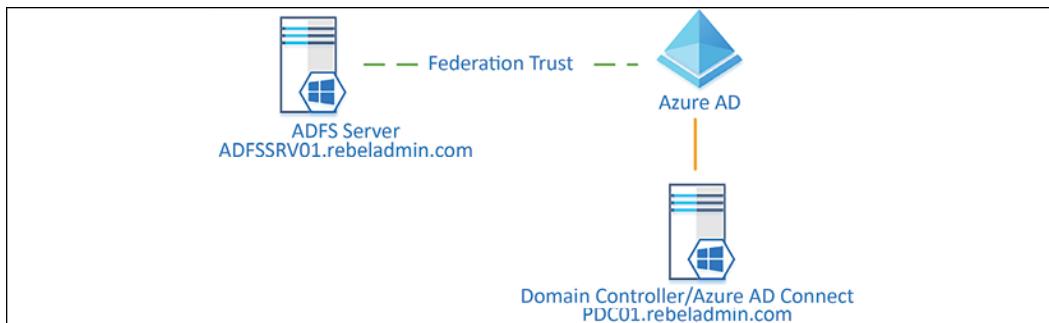


Figure 14.21: Planned demo setup

Screenshot of the Azure AD custom domain names status page:

Home > REBELADMIN >
Custom domain names ...
 Azure Active Directory

+ Add custom domain Refresh Troubleshoot Columns Got feedback?

Looking to move an on-premises application to the cloud and use Azure Active Directory Domain Services?

Search domains Add filters

Name	Status
rebeladmin.onmicrosoft.com	Available
rebeladmin.com	Verified

Figure 14.22: Azure AD custom domain status

```

PS C:\Windows\System32> Get-AdfsProperties

RunspaceId : 38d17077-6170-4352-be9c-7087e2275dae
AcceptableIdentifiers : {}
AddProxyAuthorizationRules : {}

: 38d17077-6170-4352-be9c-7087e2275dae
: {}
exists([Type ==
: http://schemas.microsoft.com/ws/2008/06/identity/claims/groupsid", Value
: "5-1-5-32-544", Issuer == "AD AUTHORITY$"] => issue(Type =
: "http://schemas.microsoft.com/authorization/claims/permit", Value =
: "true");
: c:[Type ==
: "http://schemas.microsoft.com/ws/2008/06/identity/claims/primarysid",
: Issuer == "AD AUTHORITY$"]
: => issue(store=_ProxyCredentialStore",types=("http://schemas.micr
osoft.com/authorization/claims/permit"),query="isProxyTrustManagerSid({0})
", paramvc.Value );
: c:[Type ==
: "http://schemas.microsoft.com/ws/2008/06/identity/claims/proxytrustid",
: Issuer == "SELF AUTHORITY$"]
: => issue(store=_ProxyCredentialStore",types=("http://schemas.micr
osoft.com/authorization/claims/permit"),query="isProxyTrustProvisioned({0})
", paramvc.Value );
: Data Source=spn:\.\pipe\microsoft\#adftsql\query\Initial
Catalog=adfsArtifactStore;Integrated Security=True
: urn:osis:names:tc:SAML:2.0:ac:classes:Password,
urn:osis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport,
urn:osis:names:tc:SAML:2.0:ac:classes:TLSClient,
urn:osis:names:tc:SAML:2.0:ac:classes:X509_...
: (Basic)
: True
: 2
: 365
: 20
: 5
: 720
: CN=30a49201-c09b-46c6-b238-8a1ca6f35b8a,CN=ADFS,CN=Microsoft,CN=Program
Data,DC=rebeladmin,DC=com
: 1440
: None
ContactPerson : REBELADMIN INC
DisplayName : False
IntranetUserLocalClaimsProvider : Allow
ExtendedProtectionTokenCheck : Microsoft.IdentityServer.PolicyModel.Configuration.FarmRolesConfiguration
FarmRoles : /adfs/ls/
FederationPassiveAddress : adfs.rebeladmin.com
Hostname : adfs.rebeladmin.com
HttpPort : 443
HttpsPort : 49443
Identifier : http://adfs.rebeladmin.com/adfs/services/trust
IdTokenIssuer : https://adfs.rebeladmin.com/adfs
InstalledLanguage : en-US
LogLevel : {Errors, FailureAudits, Information, Verbose...}
MonitoringInterval : 1440
NetTcpPort : 1501
NtlmOnlySupportedClientAtProxy : False
OrganizationInfo : False
PreventTokenReplays : 21600
ProxyTrustTokenLifetime : 60
ReplayCacheExpirationInterval : False
SamlMessageSignatureRequired : 5
SamlMessageDeliveryWindow : False
SignSamlAuthnRequests : 480
SsoLifetime : 129600
PersistentSsoLifetimeMins : 1440
KmsLifetimeMins : True
PersistentSsoEnabled : True

```

Figure 14.23: Existing AD FS configuration

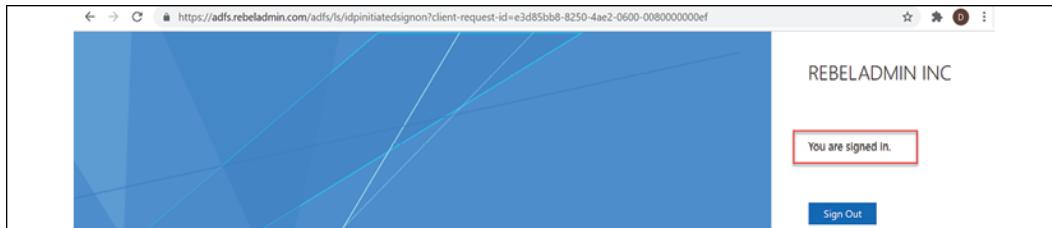


Figure 14.24: Initiated sign-in page

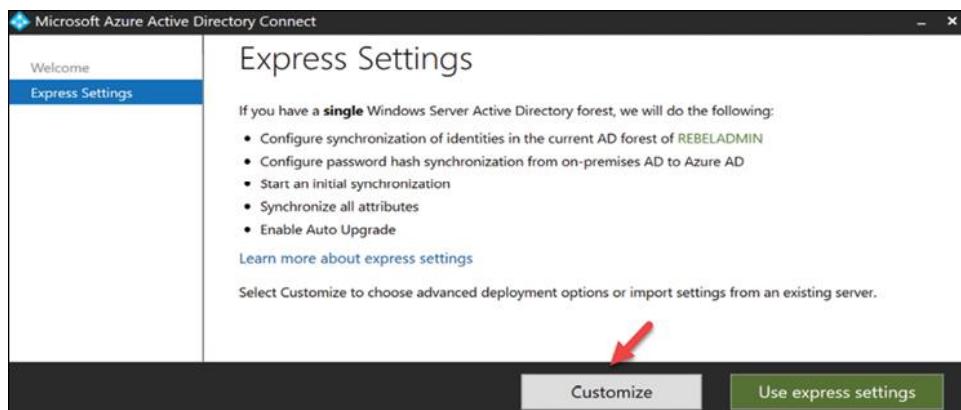


Figure 14.25: Azure AD Connect Express Settings page

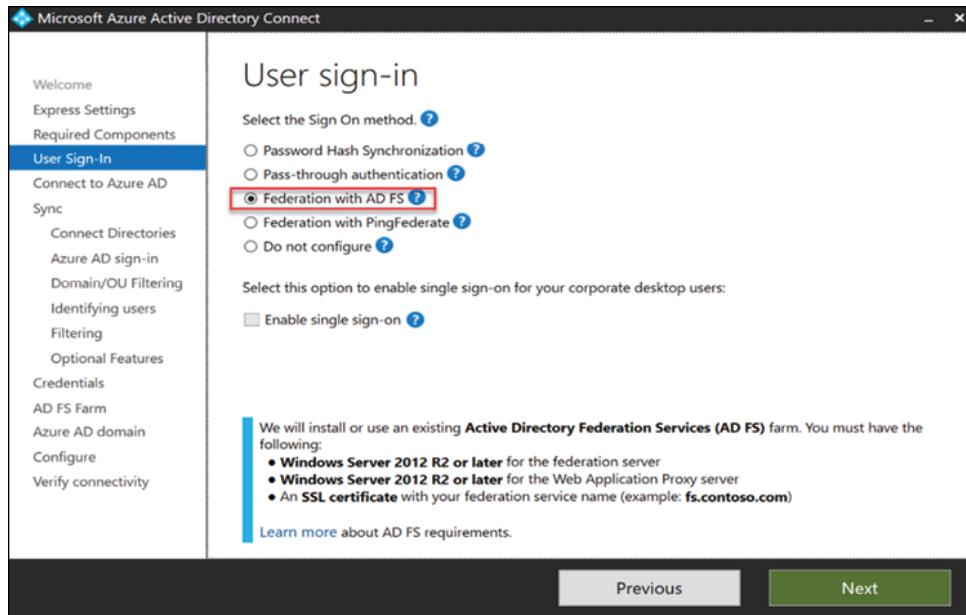


Figure 14.26: User sign-in method selection

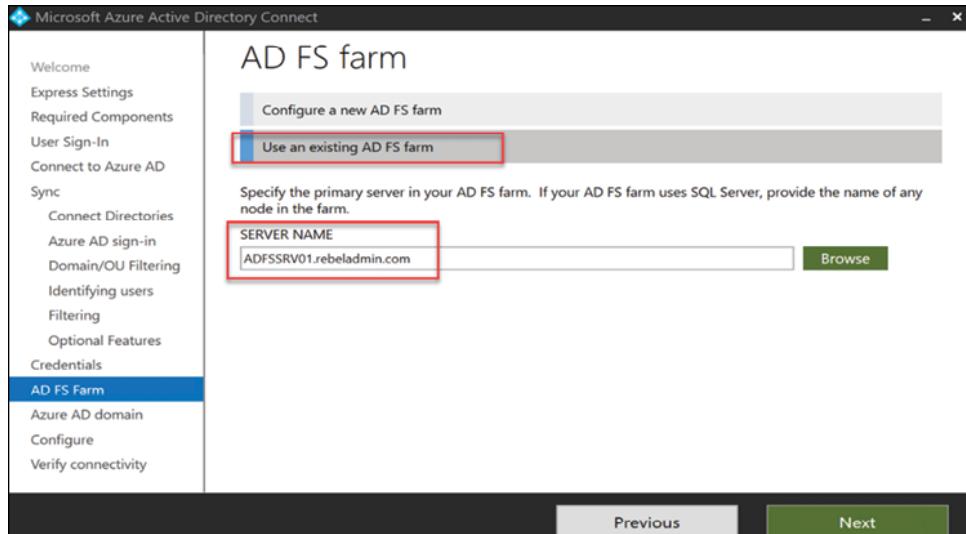


Figure 14.27: AD FS farm configuration

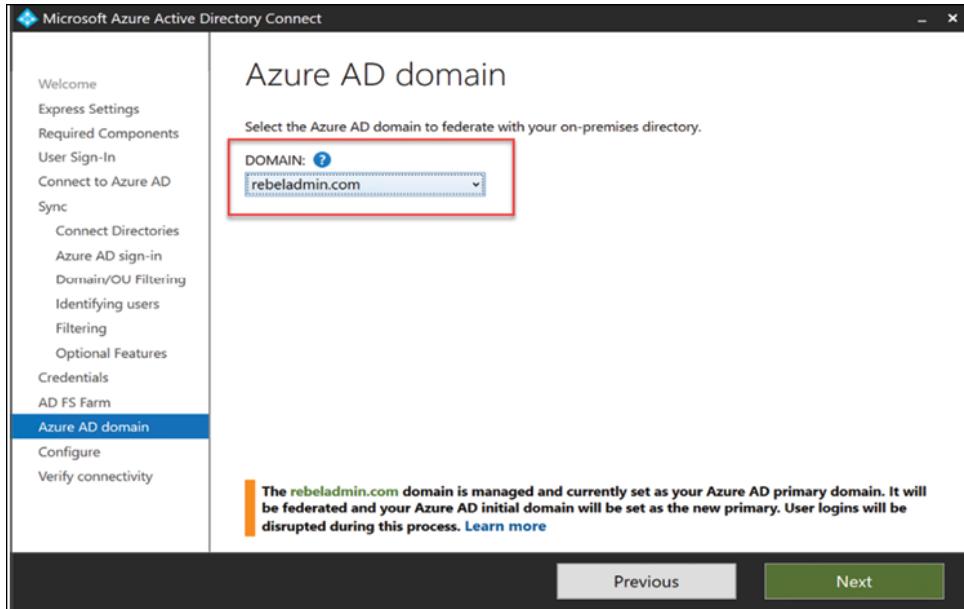


Figure 14.28: Select the domain to federate

The screenshot shows the 'Custom domain names' page in the Azure portal. The 'rebeladmin.com' domain is listed in the table. The 'Status' column shows 'Available' for 'rebeladmin.onmicrosoft.com' and 'Verified' for 'rebeladmin.com'. The 'Federated' column has a checkmark for 'rebeladmin.com'. The 'Primary' column has a checkmark for 'rebeladmin.onmicrosoft.com'.

Name	Status	Federated	Primary
rebeladmin.onmicrosoft.com	Available		✓
rebeladmin.com	Verified	✓	

Figure 14.29: Rebeladmin.com recognized as the federated domain

The screenshot shows the Microsoft sign-in page. The URL is https://sso.rebeladmin.com/adfs/ls/. The page displays the 'Sign in' form with the email address 'usera@rebeladmin.com' entered. Below the input field are links for 'Create one!' and 'Can't access your account?'. A 'Next' button is visible. At the bottom, there is a 'Sign-in options' link.

Figure 14.30: Azure AD user login page



Figure 14.31: User redirected to the AD FS login form

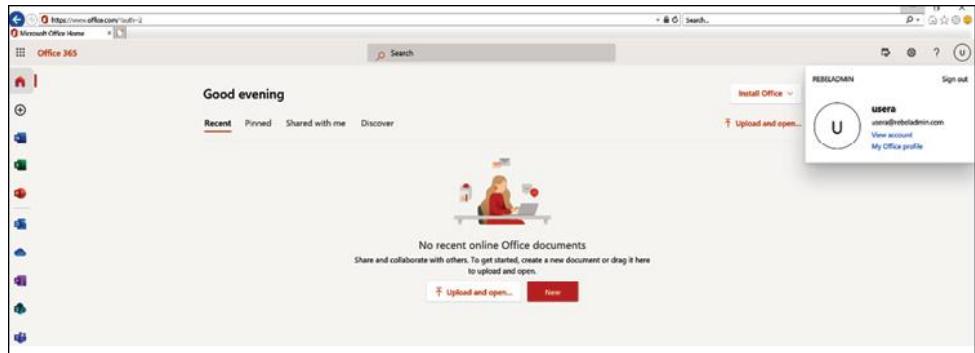


Figure 14.32: User access to the Office 365 portal

Tables

Claim type	Description
UPN	UPN of the user
Email	RFC 5322-type email address
Given name	Given name of the user
CN	CN value of the user account
Name	Name of the user
Surname	Surname of the user
Windows account name	Domain account in domain/user format
Group	Group the user belongs to
Role	Role of the user
AD FS 1.x UPN	UPN of the user when interacting with AD FS 1.x
AD FS 1.x email address	RFC 5322-type email address of the user when interacting with AD FS 1.x

Table 14.1 – AD FS claim types

Commands

Command 14.1

In order to do that, log in to the Web Application Proxy server as an administrator and execute the following command:

```
Add-WebApplicationProxyApplication
-BackendServerUrl 'https://myapp.rebeladmin.com/myapp/'
-ExternalCertificateThumbprint
'3E0ED21E43BEB1E44AD9C252A92AD5AFB8E5722E'
-ExternalUrl 'https://myapp.rebeladmin.com/myapp/'
-Name 'MyApp'
-ExternalPreAuthentication AD FS
-ADFSRelyingPartyName 'myapp.rebeladmin.com'
```

Command 14.2

First, we need to create a certificate, which will be used by the AD FS farm. This needs to run from the AD FS server:

```
$certbase64 = New-AdfsAzureMfaTenantCertificate -TenantID 05c6f80c-  
61d9-44df-bd2d-4414a983c1d4
```

The preceding command generates the new certificate. TenantID is the subscription ID you have from Azure. This can be found by running this:

```
Login-AzureRmAccount
```

The preceding command will ask for the credentials for Azure and once we provide them, it will list TenantId.

Command 14.3

Before that, we need to connect to Azure AD by using Azure PowerShell. We can do that by using the following command:

```
Connect-MsolService
```

Command 14.4

After that, we can pass the credentials by using the following command:

```
New-MsolServicePrincipalCredential -AppPrincipalId 981f26a1-7f43-403b-a875-f8b09b8cd720 -Type asymmetric -Usage verify -Value $certbase64
```

In the preceding command, AppPrincipalId defines the **Globally Unique Identifier (GUID)** for the Azure MFA client.

Links

- More information about WS-Federation can be found at <https://ibm.co/3CGyRbO>.
- Risk assessment model: <https://bit.ly/3l2Qe0m>
- ESL: <https://bit.ly/3FLLy6T>

Creating a certificate in an AD FS farm to connect to Azure MFA

First, we need to create a certificate, which will be used by the AD FS farm. This needs to run from the AD FS server:

```
$certbase64 = New-AdfsAzureMfaTenantCertificate -TenantID 05c6f80c-61d9-44df-bd2d-4414a983c1d4
```

The preceding command generates the new certificate. TenantID is the subscription ID you have from Azure. This can be found by running this:

```
Login-AzureRmAccount
```

The preceding command will ask for the credentials for Azure and once we provide them, it will list TenantId:

```
Environment      : AzureCloud
Account         : dcadmin@REBELADMIN.onmicrosoft.com
TenantId        : 05c6f80c-61d9-44df-bd2d-4414a983c1d4
SubscriptionId   :
SubscriptionName :
CurrentStorageAccount :
```

Figure 14.14: Azure TenantId

This will create a certificate under **Certificates (Local Computer)**:

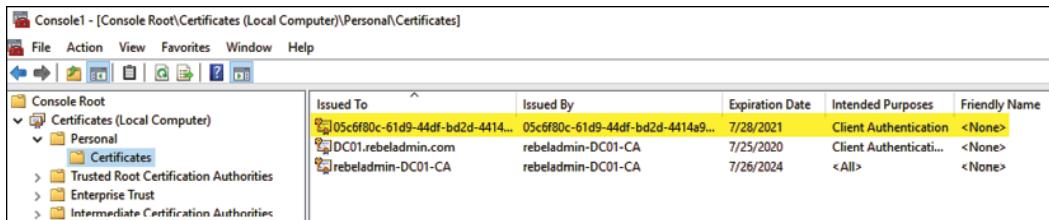


Figure 14.15: New certificate

Enabling AD FS servers to connect with the Azure MFA client

Now, we have the certificate, but we need to tell the Azure MFA client to use it as a credential to connect with AD FS.

Before that, we need to connect to Azure AD by using Azure PowerShell. We can do that by using the following command:

```
Connect-MsolService
```

Then, it will prompt you for your login and use your Azure Global Administrator account to connect.

After that, we can pass the credentials by using the following command:

```
New-MsolServicePrincipalCredential -AppPrincipalId 981f26a1-7f43-403b-a875-f8b09b8cd720 -Type asymmetric -Usage verify -Value $certbase64
```

In the preceding command, `AppPrincipalId` defines the **Globally Unique Identifier (GUID)** for the Azure MFA client.

Enabling the AD FS farm to use Azure MFA

The next step of the configuration is to enable the AD FS farm to use Azure MFA. This can be done by using the following command:

```
Set-AdfsAzureMfaTenant -TenantId 05c6f80c-61d9-44df-bd2d-4414a983c1d4 -ClientId 981f26a1-7f43-403b-a875-f8b09b8cd720
```

In the preceding command, `TenantId` refers to the Azure tenant ID and `ClientId` represents the Azure MFA client's GUID.

Once the command successfully runs, we need to restart the AD FS service on each server in the farm:

```
PS C:\Users\administrator.REBELADMIN\Desktop> Set-AdfsAzureMfaTenant -TenantId 05c6f80c-61d9-44df-bd2d-4414a983c1d4 -ClientId 981f26a1-7f43-403b-a875-f8b09b8cd720
WARNING: PS0177: The authentication provider configuration data was successfully updated. Before your changes take effect, you must restart the AD FS Windows Service on each server in the farm.
PS C:\Users\administrator.REBELADMIN\Desktop>
```

Figure 14.16: Enabling Azure MFA for AD FS

Enabling Azure MFA for authentication

The last step of the configuration is to enable Azure MFA globally for the AD FS server.

In order to do that, log in to the AD FS server as the Enterprise Admin. Then, go to **Server Manager | Tools | AD FS Management**.

Then, in the console, navigate to **Service | Authentication Methods**. Then, in the **Actions** panel, click on **Edit Primary Authentication Method**:

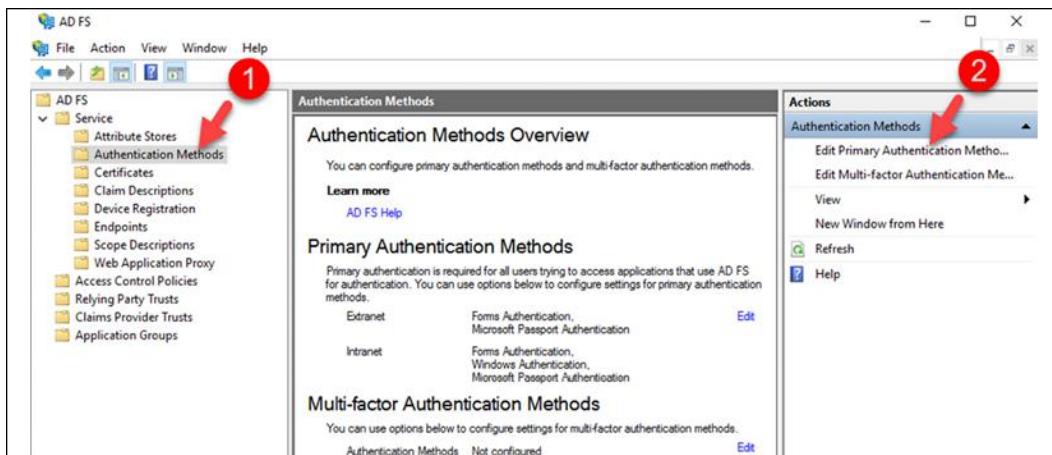


Figure 14.17: AD FS Authentication Methods

This opens up the window to configure global authentication methods. It has two tabs, and we can see **Azure MFA** on both. If Azure MFA is used as a primary method by removing other options, then AD FS will not ask for logins and will use MFA as the only authentication method.

Its operation boundaries can be set to intranet or extranet:

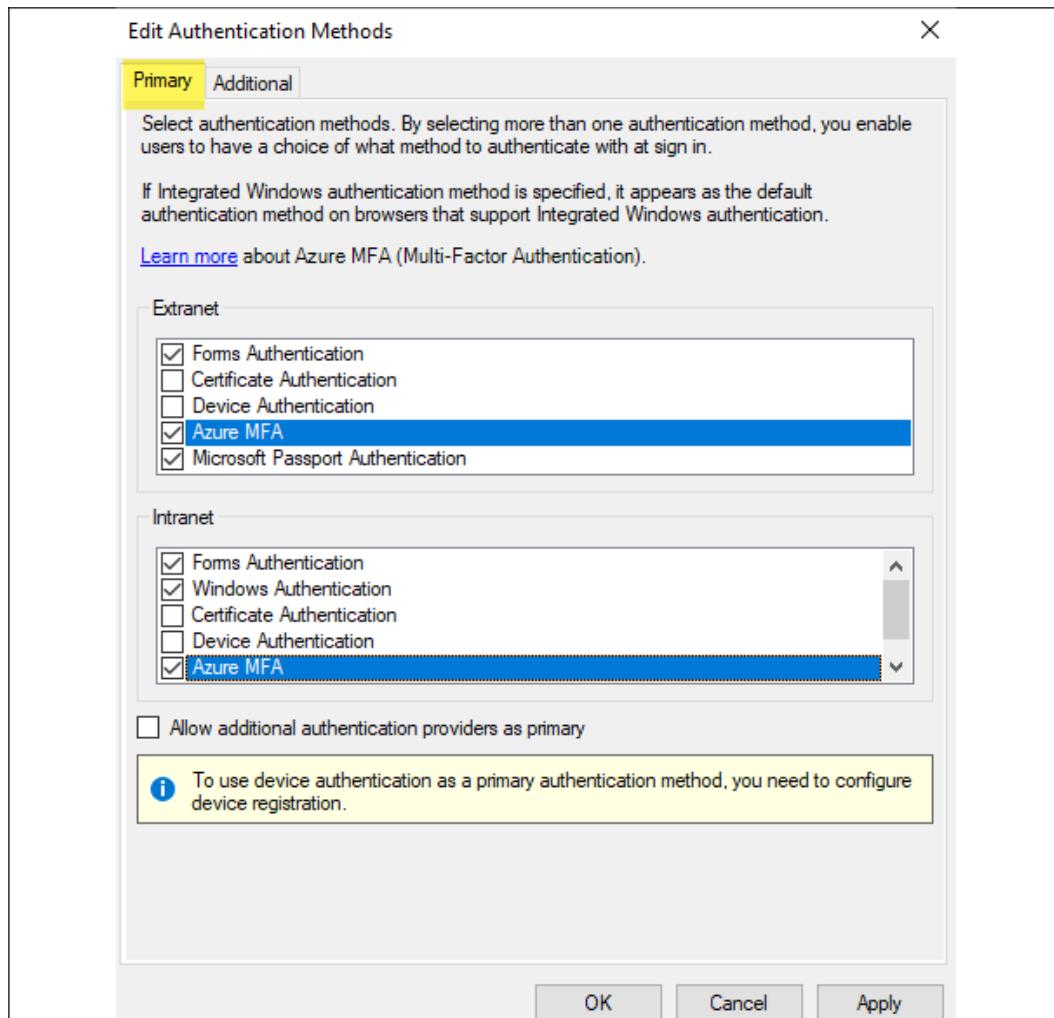


Figure 14.18: AD FS Primary authentication methods

Another option is to select MFA as the secondary authentication method:

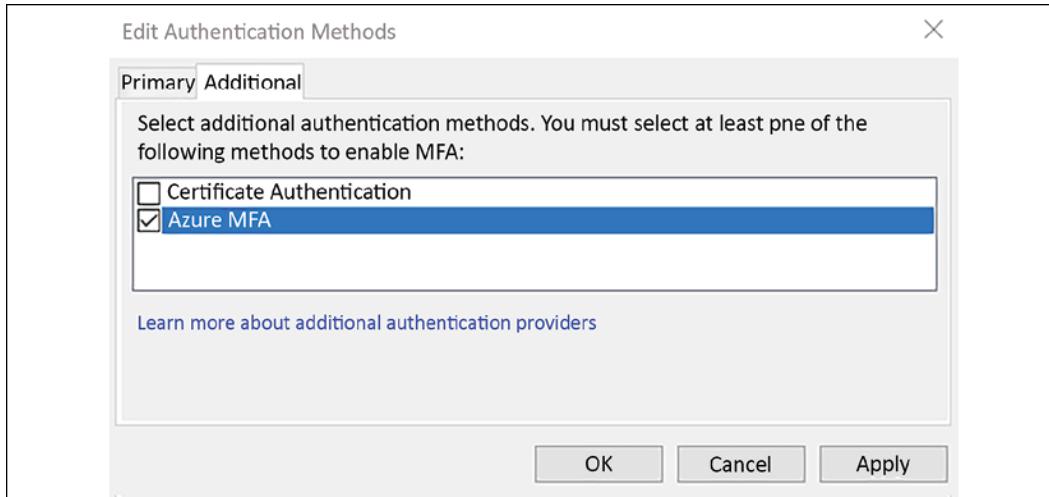


Figure 14.19: AD FS Additional authentication methods

This finishes the Azure MFA integration, and users can use MFA based on the options selected in the preceding wizards.

Installing the AD FS role

Before installation, the SSL certificate for adfs.rebeladmin.com needs to be installed in the computer account as it is required during the AD FS installation. This can be checked using the following command:

```
dir Cert:\LocalMachine\My
```

The AD FS server should be a member server of the domain and should log in as the domain administrator or the Enterprise Admin to do the installation.

The next step is to install the AD FS role service, which can be done by using the following PowerShell command:

```
Install-WindowsFeature ADFS-Federation -IncludeManagementTools
```

The following screenshot displays the output of the preceding command:

```
Administrator: PowerShell 7 (x64)
PowerShell 7.1.4
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Users\dfrancis> Install-WindowsFeature ADFS-Federation -IncludeManagementTools

Success Restart Needed Exit Code      Feature Result
----- ----- ----- -----
True   No       Success          {Active Directory Federation Services}
WARNING: To finish configuring this server for the federation server role using Windows PowerShell, see http://go.microsoft.com/fwlink/?LinkId=224868.

PS C:\Users\dfrancis>
```

Figure 14.6: Install the AD FS role

Once this is completed, we need to configure the AD FS server. Let's use the following configuration for the demo setup:

```
Import-Module ADFS
$credentials = Get-Credential
Install-AdfsFarm `-
-CertificateThumbprint:"938E369FA88B2F884A5BBC495F2338BE9FA0E0BB" `-
-FederationServiceDisplayName:"REBELADMIN INC" `-
-FederationServiceName:"adfs.rebeladmin.com" `-
-ServiceAccountCredential $credentials
```

In this setup, we are using WID for AD FS, so there is no need for SQL configuration. In the preceding command, `CertificateThumbprint` specifies the SSL certificate (adfs.rebeladmin.com), and `FederationServiceDisplayName` specifies the display name of the federation service. `FederationServiceName` is the service name, and it should match the SSL we used.

`ServiceAccountCredential` is used to define the service account details for the AD FS setup. In the end, the system needs to be restarted to apply the configuration:

```
PS C:\Users\dfrancis.REBELADMIN> Import-Module ADFS
WARNING: Module ADFS 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.
PS C:\Users\dfrancis.REBELADMIN> $credentials = Get-Credential

PowerShell credential request
Enter your credentials.
User: rebeladmin.com\dfrancis
Password for user rebeladmin.com\dfrancis: *****

PS C:\Users\dfrancis.REBELADMIN> Install-AdfsFarm `-
>> -CertificateThumbprint:"69F65A89906485DFA8F9DF8CC239508BF3256673" `-
>> -FederationServiceDisplayName:"REBELADMIN INC" `-
>> -FederationServiceName:"adfs.rebeladmin.com" `-
>> -ServiceAccountCredential $credentials
WARNING: A machine restart is required to complete ADFS service configuration. For more information, see: https://go.microsoft.com/fwlink/?LinkId=798725
WARNING: The SSL certificate subject alternative names do not support host name 'certauth.adfs.rebeladmin.com'. Configuring certificate authentication binding on port '49443' and hostname 'adfs.rebeladmin.com'.

RunspaceId : 1dbbc54-0bc4-4994-ac17-602ea129479
Message    : The configuration completed successfully.
Context    : DeploymentSucceeded
Status     : Success
```

Figure 14.7: Configure the AD FS role

The error about the alternative SSL name, certauth.adfs.rebeladmin.com, regards the certificate authentication. Before Windows Server 2016, this was an issue as the system didn't support different bindings for certificate authentication and device authentication on the same host. The default port 443 was used by the device authentication and couldn't have multiple bindings on the same channel.

In Windows Server 2016/2016/2019/2022, this is possible and now, it supports two methods. The first option is to use the same host (`adfs.rebeladmin.com`) with different ports (443 and 49443). The second option is to use different hosts (`adfs.rebeladmin.com` and `certauth.adfs.rebeladmin.com`) with the same port (443). This requires an SSL certificate to support `certauth.adfs.rebeladmin.com` as an alternate subject name.

Once the reboot completes, we can check whether the installation was successful by using the following command:

```
Get-WinEvent "AD FS/Admin" | Where-Object {$_.ID -eq "100"} | fl
```

This will print the content of event 100, which confirms the successful AD FS installation.

Installing WAP

The next step of the configuration is to install WAP. This doesn't need to be a domain-joined server and should be placed on the perimeter network. Before the installation process, install the required SSL certificates. In my demo, it is for *.rebeladmin.com. We can verify this by using this:

```
dir Cert:\LocalMachine\My
```

Before proceeding, we also need to check whether a server can resolve to adfs.rebeladmin.com as WAP needs to connect to AD FS.

Once everything is confirmed, we can install the WAP role:

```
Install-WindowsFeature Web-Application-Proxy -IncludeManagementTools
```

Once it's completed, we can proceed with the configuration by using the following:

```
$credentials = Get-Credential  
Install-WebApplicationProxy  
-FederationServiceName "adfs.rebeladmin.com"  
-FederationServiceTrustCredential $credentials  
-CertificateThumbprint "3E0ED21E43BEB1E44AD9C252A92AD5AFB8E5722E"
```

In the preceding commands, FederationServiceName is used to define the AD FS service name, and it needs to match the name provided on the AD FS setup. FederationServiceTrustCredential is used to provide an account, which is authorized to register a new proxy server with AD FS. The account that is used here should have permissions to manage AD FS.

The CertificateThumbprint parameter is used to define the certificate for WAP. In our demo, it's the *.rebeladmin.com certificate. At the end of the configuration, we need to restart the system to apply the changes.

Once the reboot is completed, we can confirm the health of the configuration using the following event log in the AD FS server:

```
Get-WinEvent "AD FS/Admin" | Where-Object {$_.ID -eq "396"} | fl
```

Chapter 15

Links

AIP implementation is a vast topic that is beyond the scope of this chapter. I have written a series of blog posts covering the implementation of AIP:

- *Step-by-Step Guide: Protect confidential data using Azure Information Protection:*
<https://bit.ly/30NxJG1>
- *Step-by-Step Guide: Automatic Data Classification via Azure Information Protection:*
<https://bit.ly/30ZZGe5>
- *Step-by-Step Guide: On-premise Data Protection via Azure Information Protection Scanner:*
<https://bit.ly/3nKfiei>
- *Step-by-Step Guide: How to protect confidential emails using Azure Information Protection?:*
<https://bit.ly/3I4LU0x>
- *Step-by-Step Guide: How to track shared documents using Azure Information Protection?:*
<https://bit.ly/3FYNGZt>

Figures/Tables

Table 15.1

Requirements	Application	Protected file types
Protect confidential files	Microsoft Outlook Windows (version 2003 onward) Office for macOS 2016: Word Excel PowerPoint PDF Image processing XPS Viewer Gaaiho Doc GigaTrust Desktop PDF Client for Adobe Foxit PDF Reader Nitro PDF Reader Siemens JT2Go	.doc, .docm, .docx, .dot, .dotm, .dotx, .potm, .potx, .pps, .ppsm, .ppsx, .ppt, .pptm, .txt, .xml, .jpg, .jpeg, .pdf, .png, .tif, .tiff, .bmp, .gif, .jpe, .jfif, .jt, .xps
Protect confidential emails	Microsoft Outlook Windows (version 2003 and newer) Office for macOS 2016	.msg

	Microsoft Exchange 2007 SP1	
Protect content on the intranet	Microsoft SharePoint 2007 and newer	.doc, .docm, .docx, .dot, .dotm, .dotx, .potm, .potx, .pps, .ppsm, .ppsx, .ppt, .pptm, .txt, .xml, .jpg, .jpeg, .pdf, .png, .tif, .tiff, .bmp, .gif, .jpe, .jfif, .jt, .xps
Protect mobile data	Microsoft Word Microsoft Word app Microsoft Excel app Microsoft Outlook app Microsoft PowerPoint app WordPad Office for macOS 2016 Word Online TITUS Docs Foxit Reader	.doc, .docm, .docx, .dot, .dotm, .dotx, .potm, .potx, .pps, .ppsm, .ppsx, .ppt, .pptm, .txt, .xml, .jpg, .jpeg, .pdf, .png, .tif, .tiff, .bmp, .gif, .jpe, .jfif, .jt, .msg, .xps

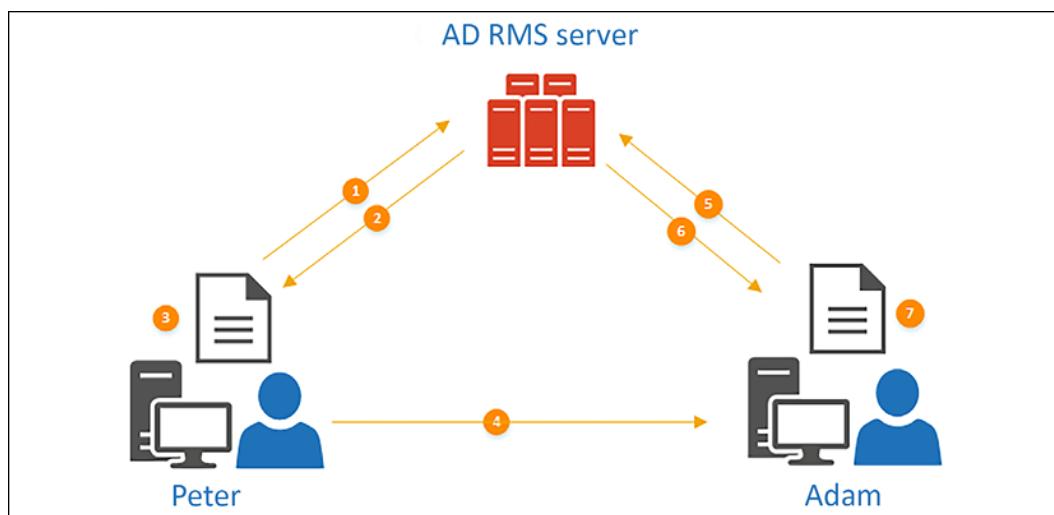


Figure 15.1: AD RMS in action

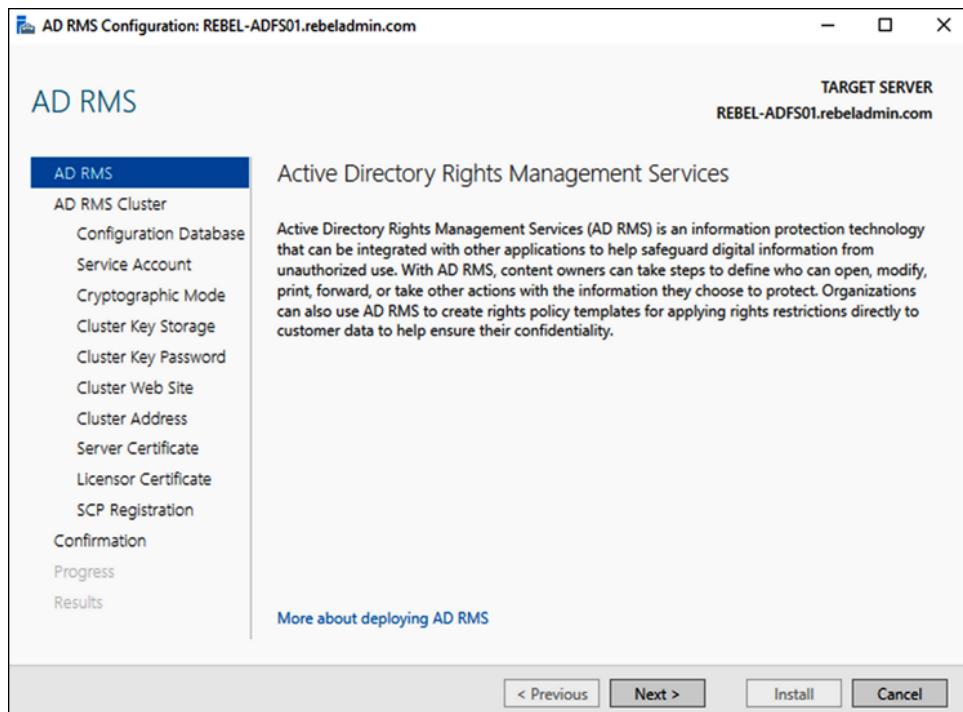


Figure 15.2: AD RMS configuration wizard

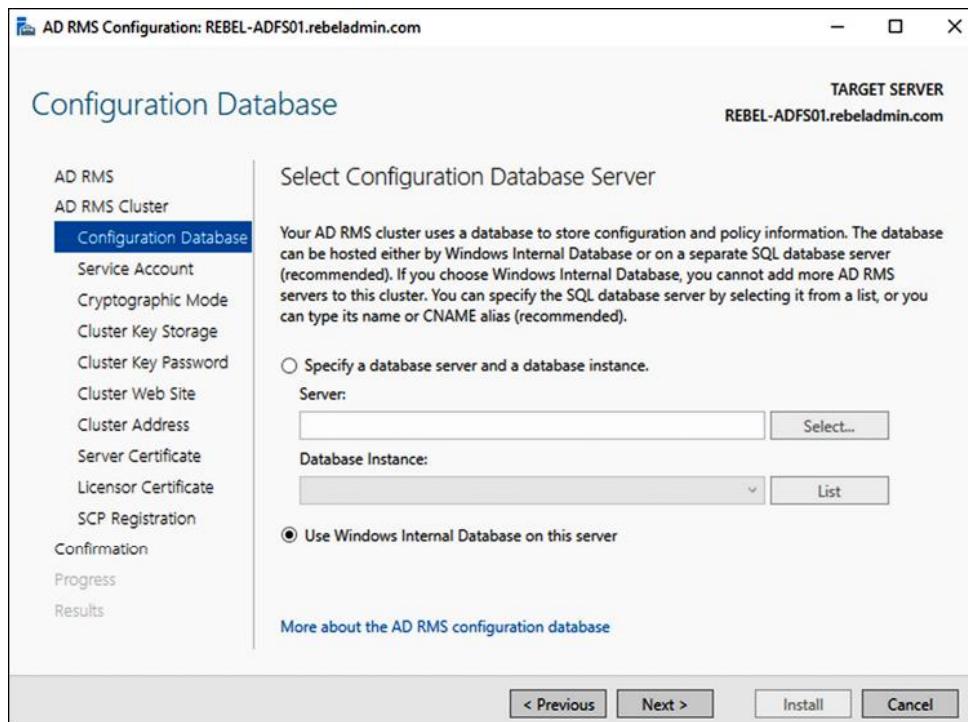


Figure 15.3: Creating WID for AD RMS

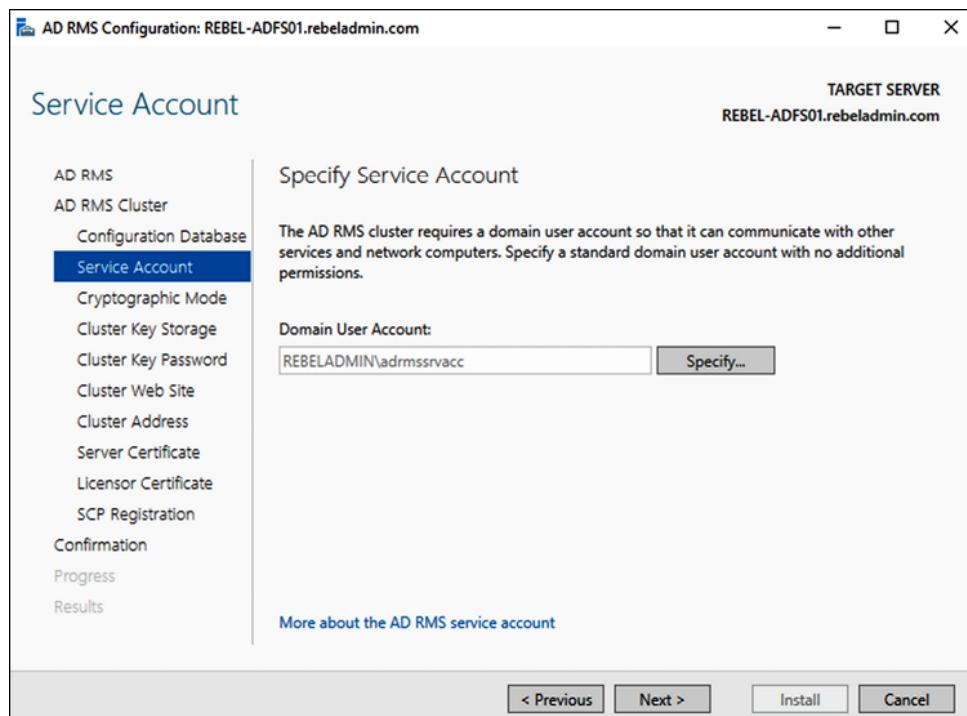


Figure 15.4: Defining the AD RMS service account

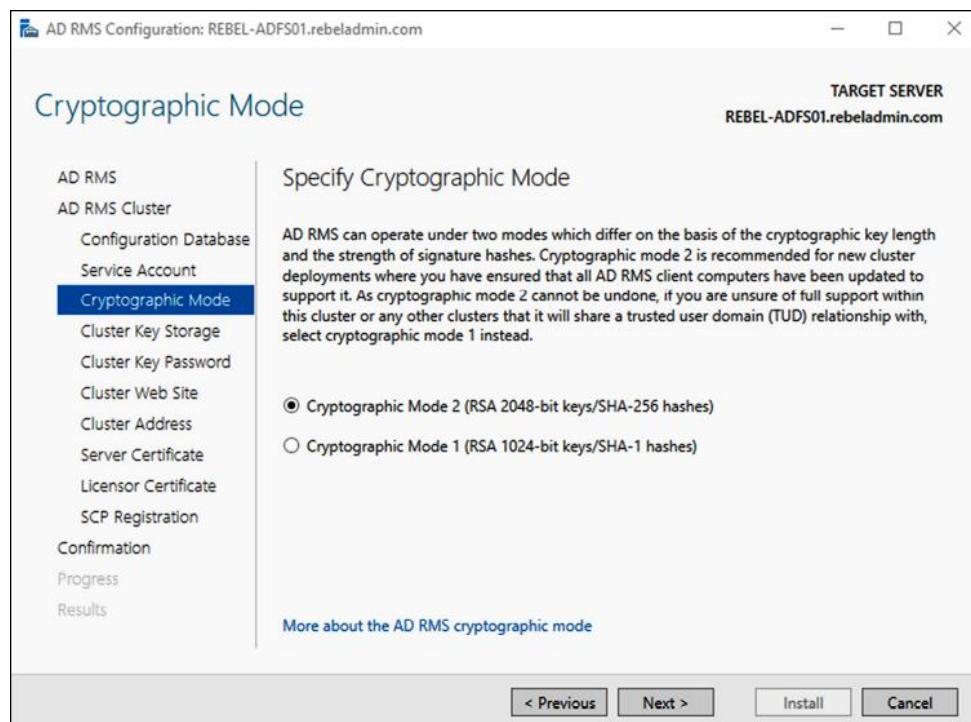


Figure 15.5: Defining the AD RMS cryptographic mode

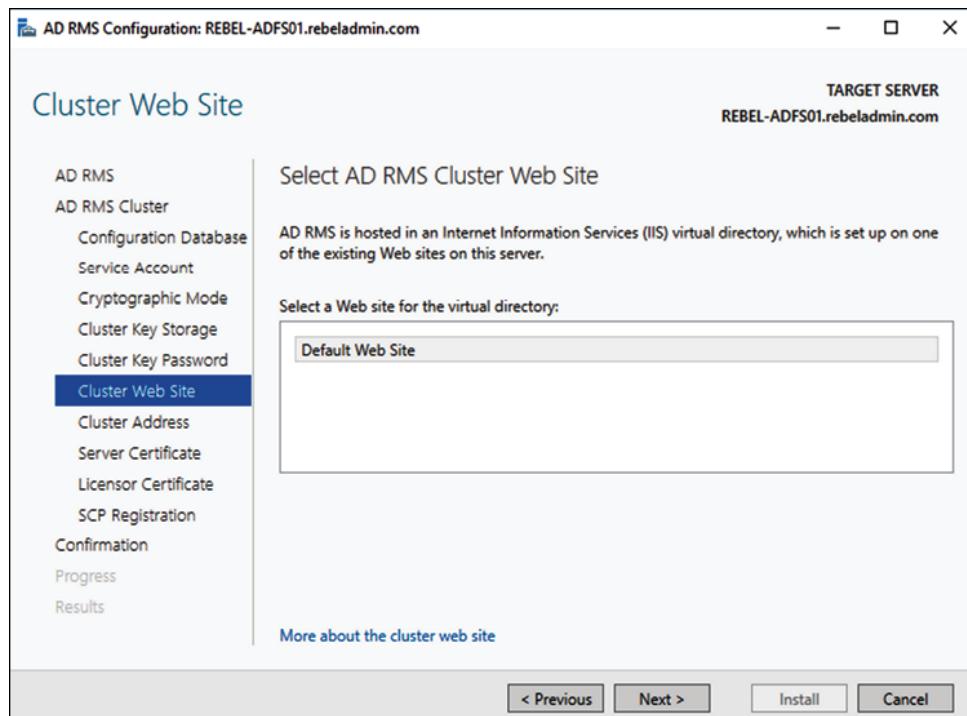


Figure 15.6: AD RMS cluster website settings

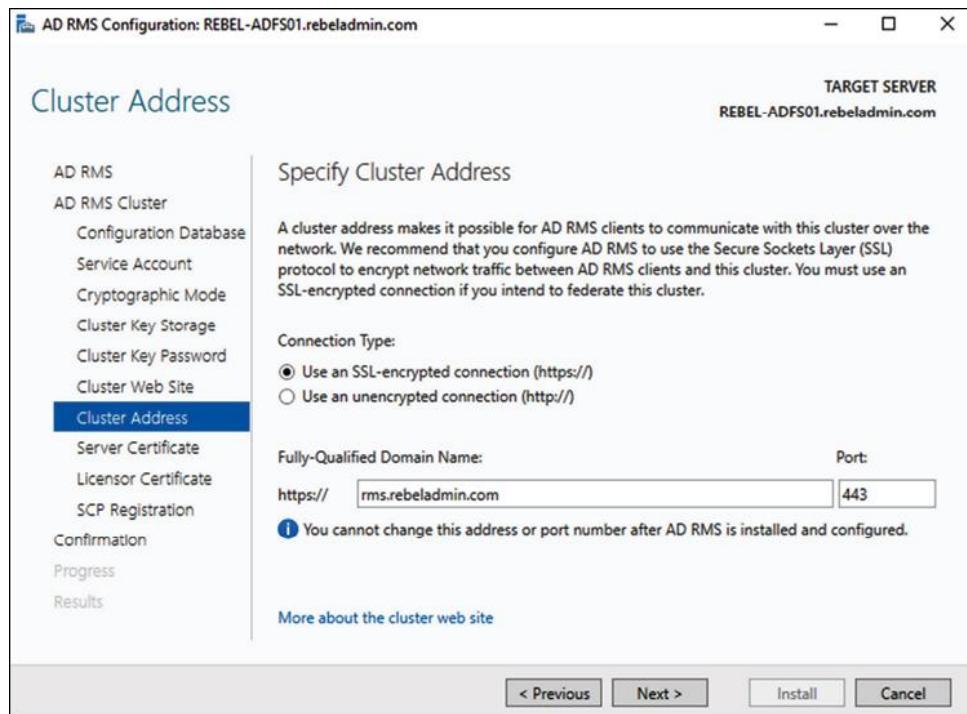


Figure 15.7: AD RMS cluster address

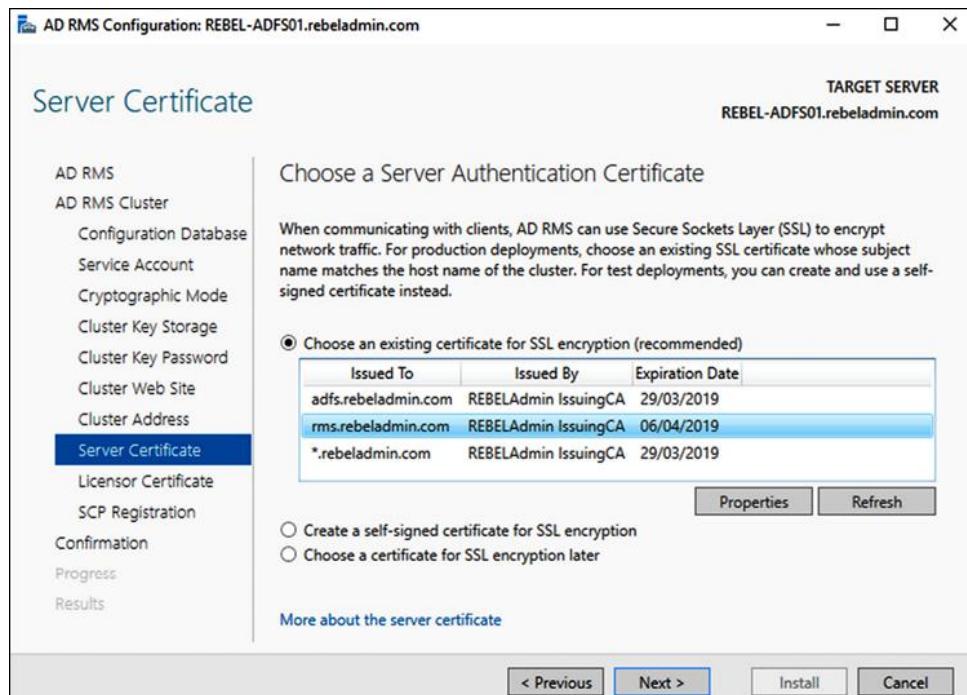


Figure 15.8: Selecting the server authentication certificate

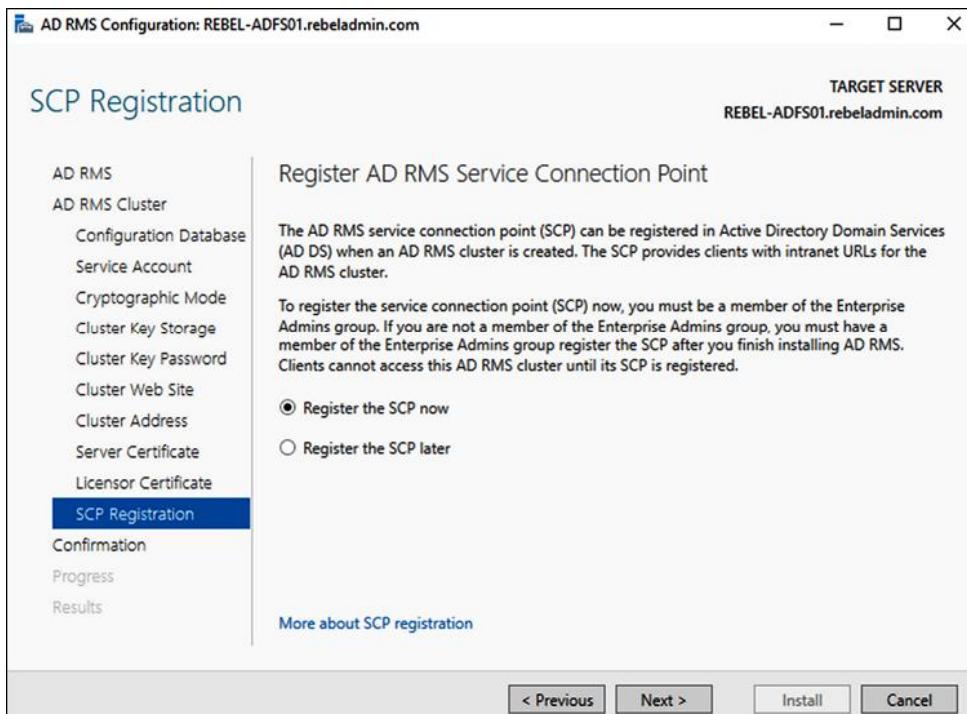


Figure 15.9: Registering an AD RMS SCP

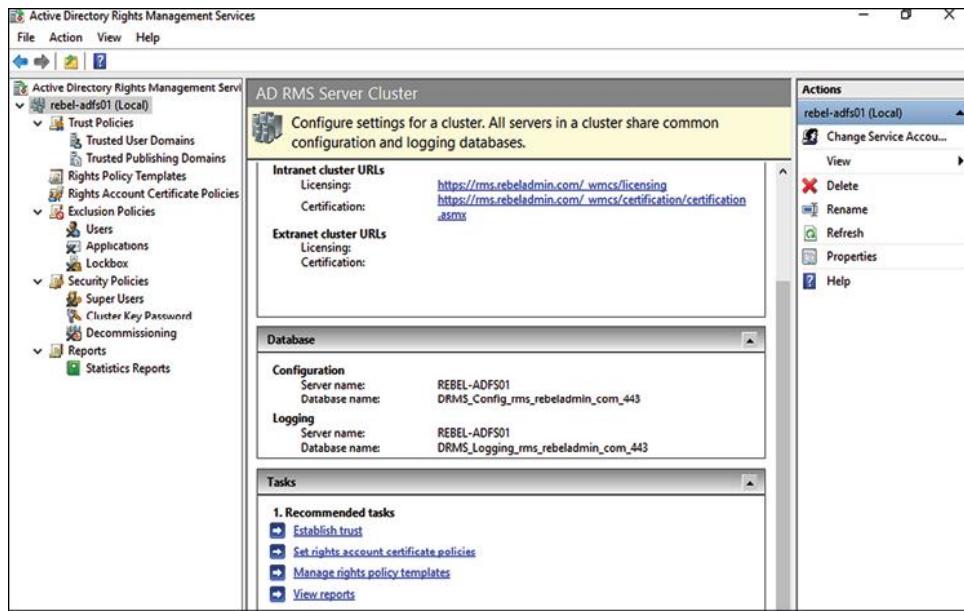


Figure 15.10: AD RMS server cluster

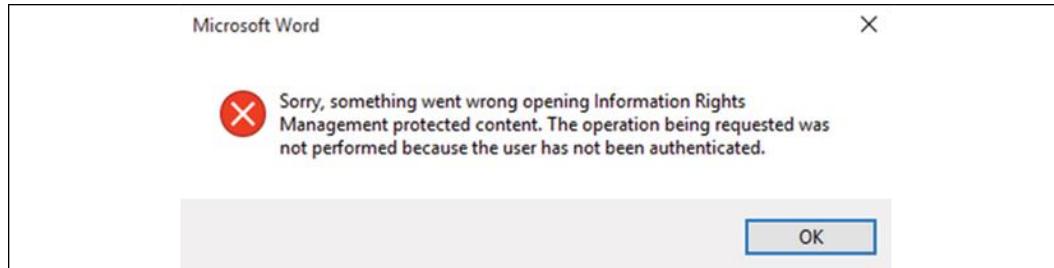


Figure 15.11: Service error due to trusted sites

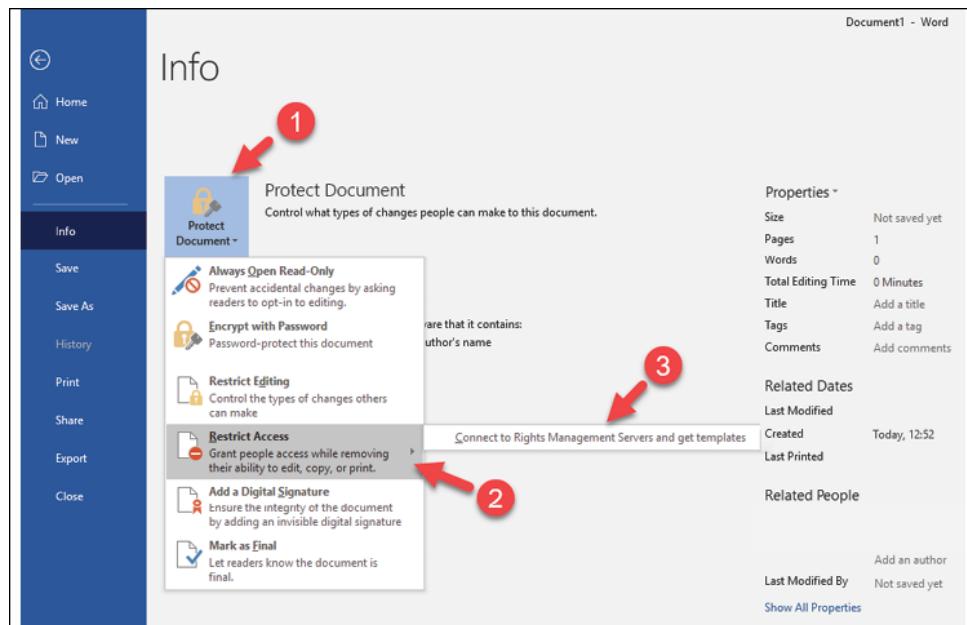


Figure 15.12: Connecting to the AD RMS server to get templates

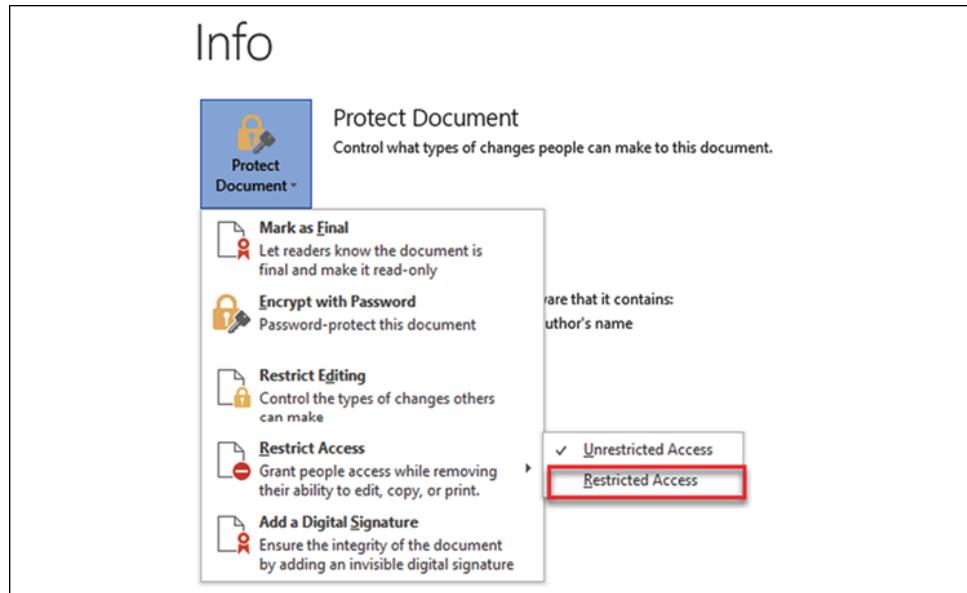


Figure 15.13: Protecting a Word document with AD RMS

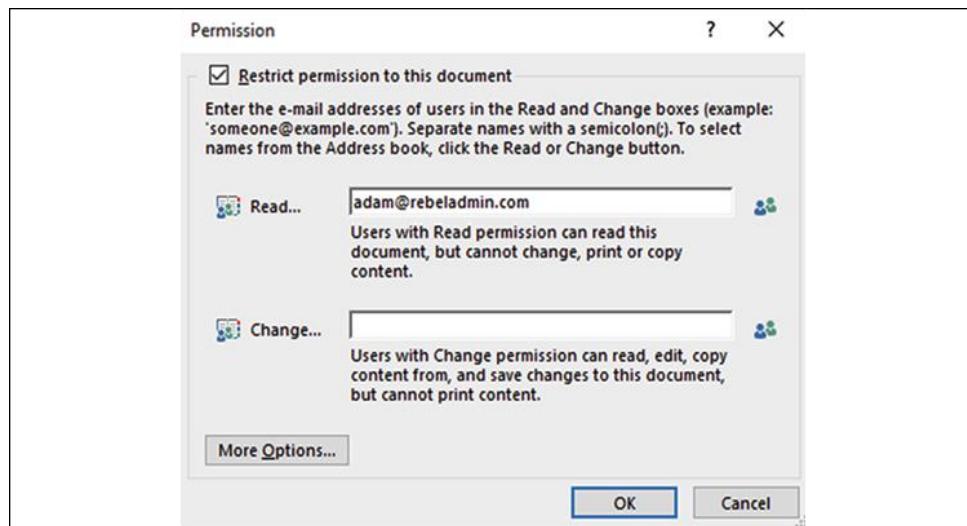


Figure 15.14: Document permissions

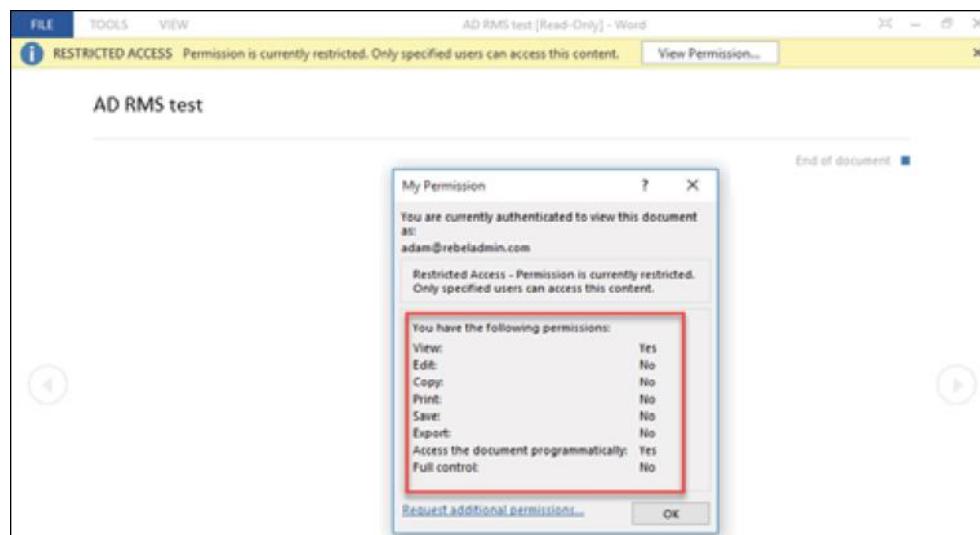


Figure 15.15: Viewing permissions

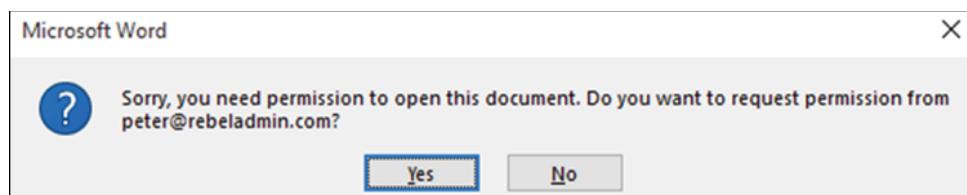


Figure 15.16: Error message

Age 4–5	Age 5–6	Age 6 – 7
Reading at school School Year: Reception Band: Lilac Level: 1 Band: Pink Level: 1+ Band: Red Level: 2 Band: Yellow Level: 3	Reading at school School Year: 1 Band: Blue Level: 4 Band: Green Level: 5 Band: Orange Level: 6	Reading at school School Year: 2 Band: Turquoise Level: 7 Band: Purple Level: 8 Band: Gold Level: 9 Band: White Level: 10 Band: Lime Level: 11
Reading at home <i>Read with Oxford: Stage 2</i> eBook library: eBooks for 4–5 year olds	Reading at home <i>Read with Oxford: Stage 3 & Stage 4</i> eBook library: eBooks for 5–6 year olds	Reading at home <i>Read with Oxford: Stage 4, Stage 5 & Stage 6</i> eBook library: eBooks for 6–7 year olds

Figure 15.17: Oxford reading levels

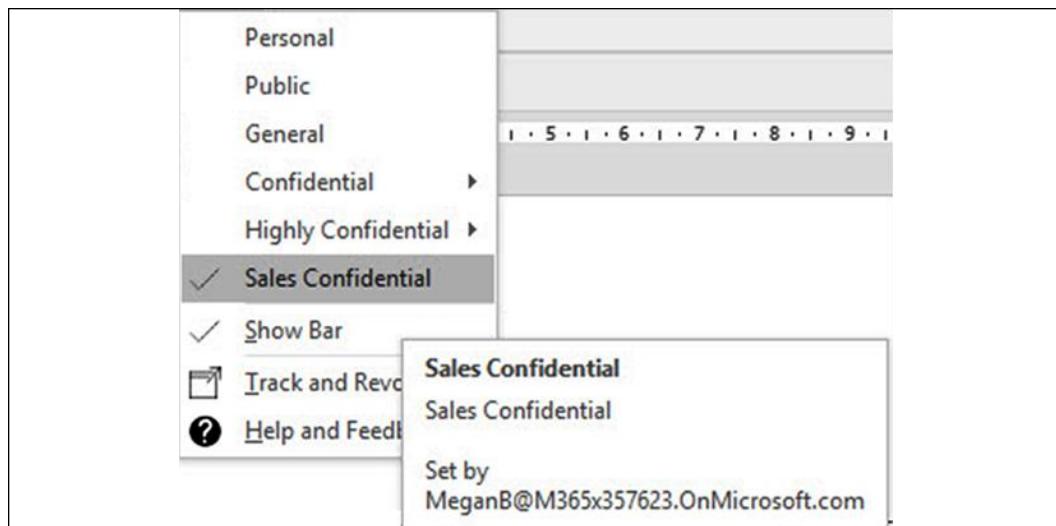


Figure 15.18: Example of labels



Figure 15.19: Automated classification

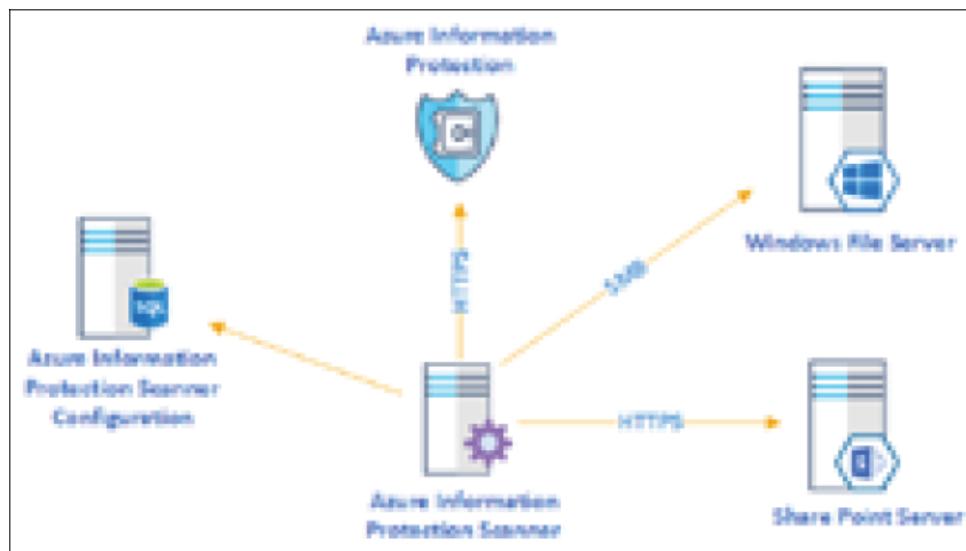


Figure 15.20: AIP scanner

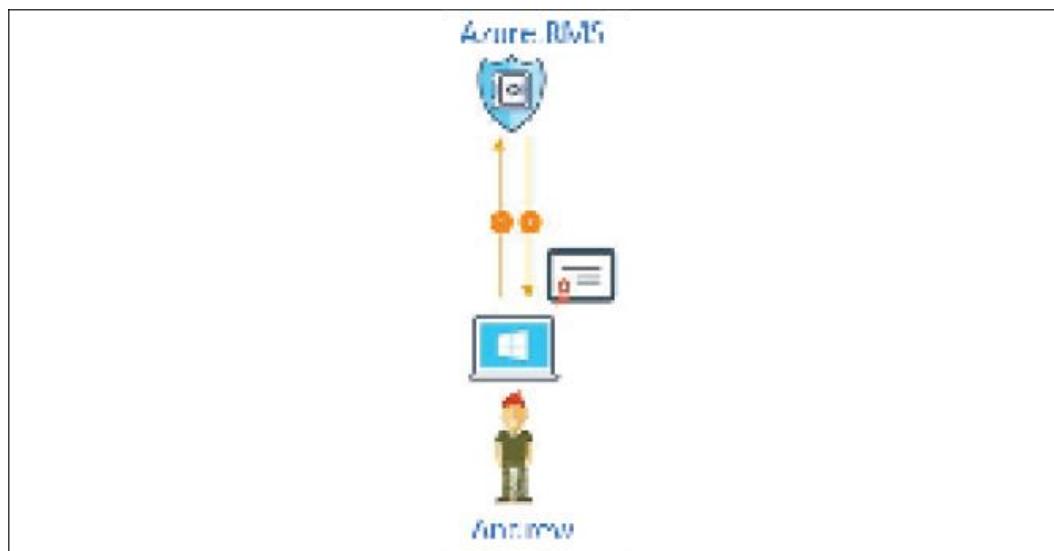


Figure 15.21: Initial communication with Azure RMS

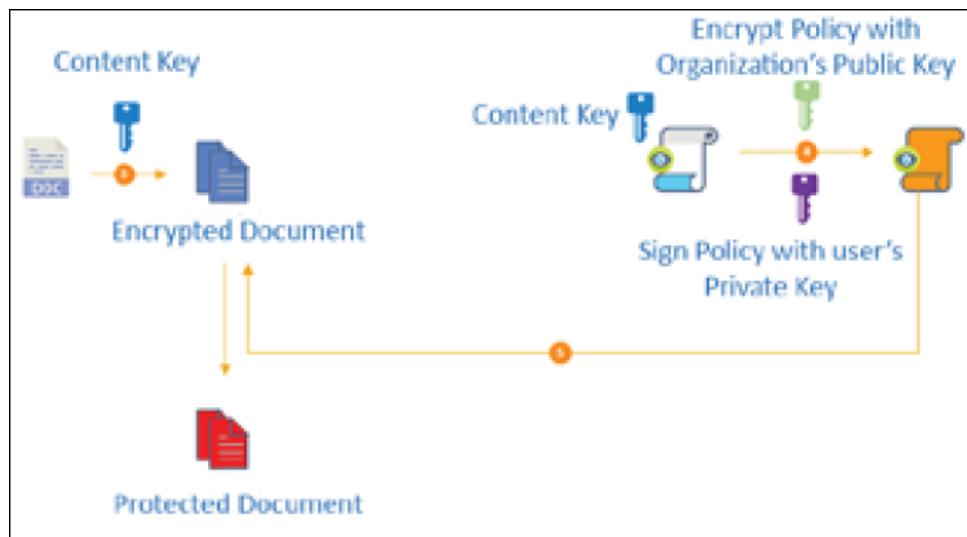


Figure 15.22: Preparing for document protection

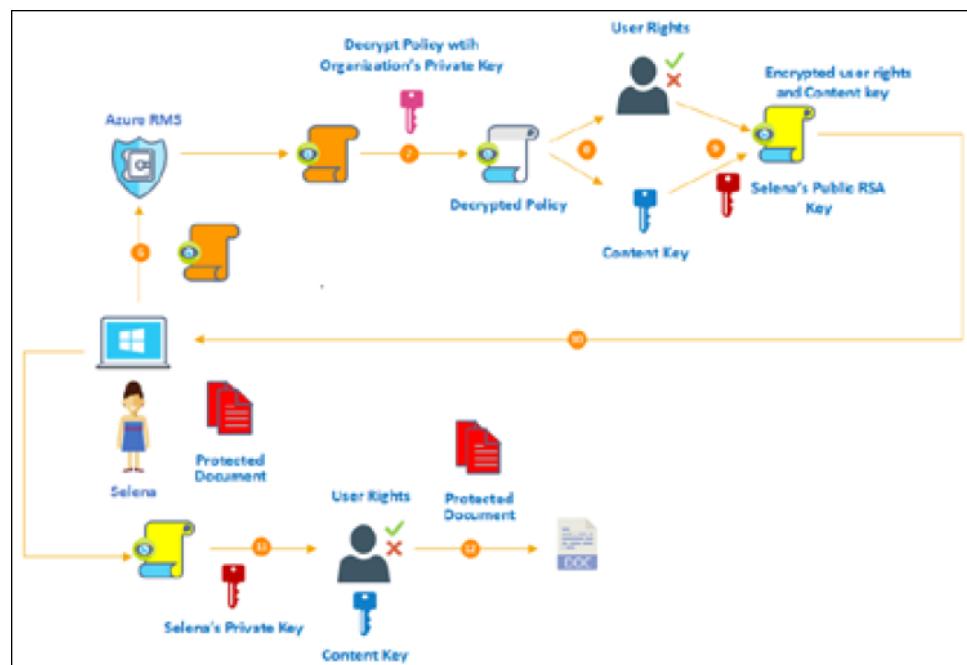


Figure 15.23: Data protection with Azure RMS

Commands

Command 15.1

```
Install-WindowsFeature ADRMS -IncludeManagementTools
```

Chapter 16

Figures



Figure 16.1: Authentication by using a secret

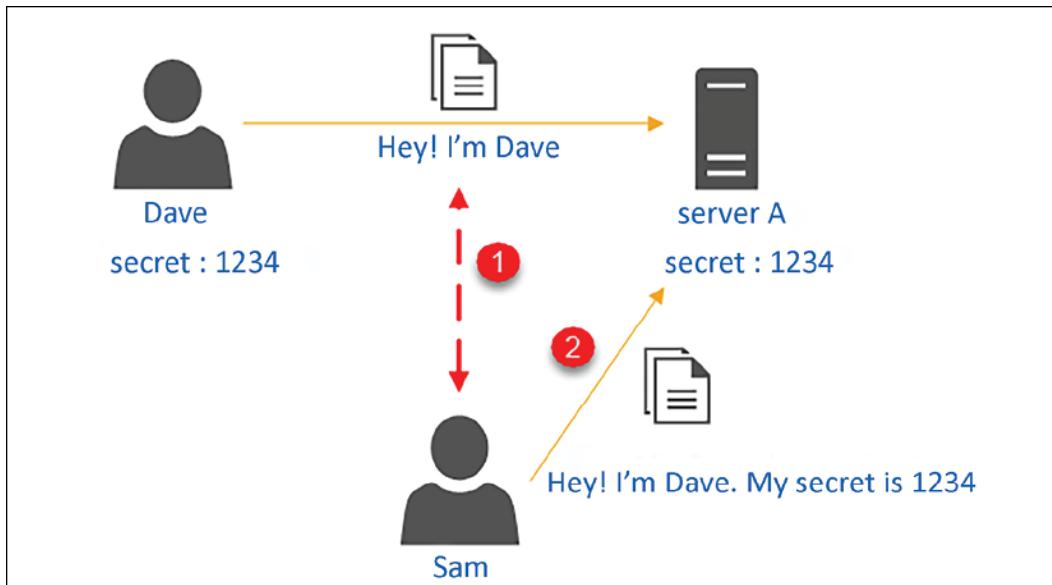


Figure 16.2: Man-in-the-middle attack

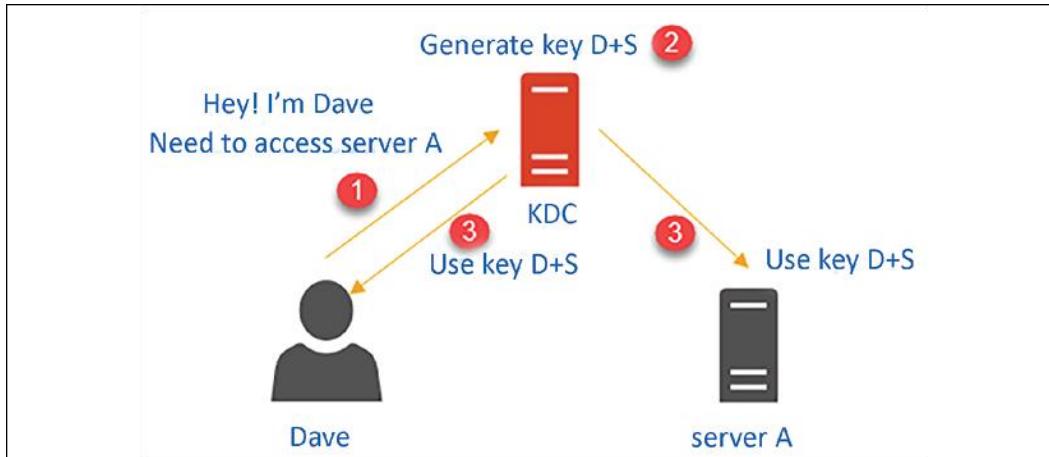


Figure 16.3: KDC in action

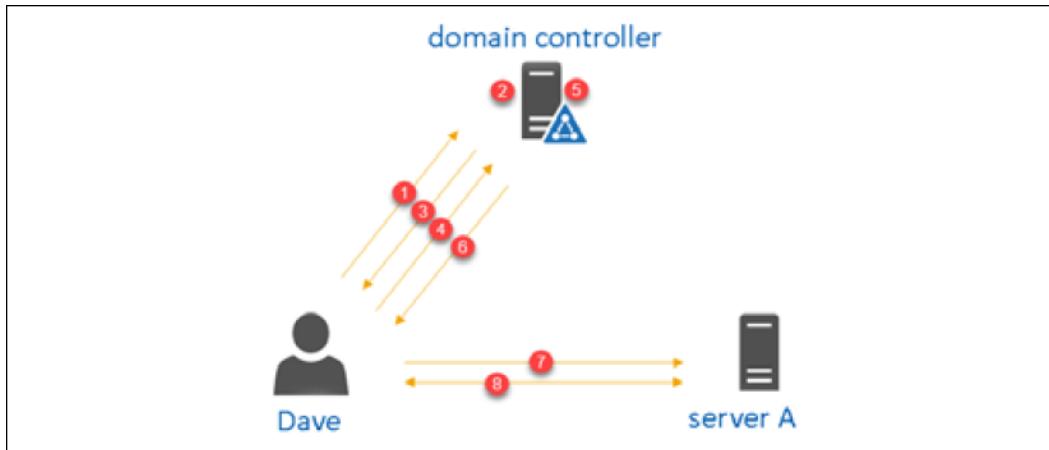


Figure 16.4: Authentication process in AD

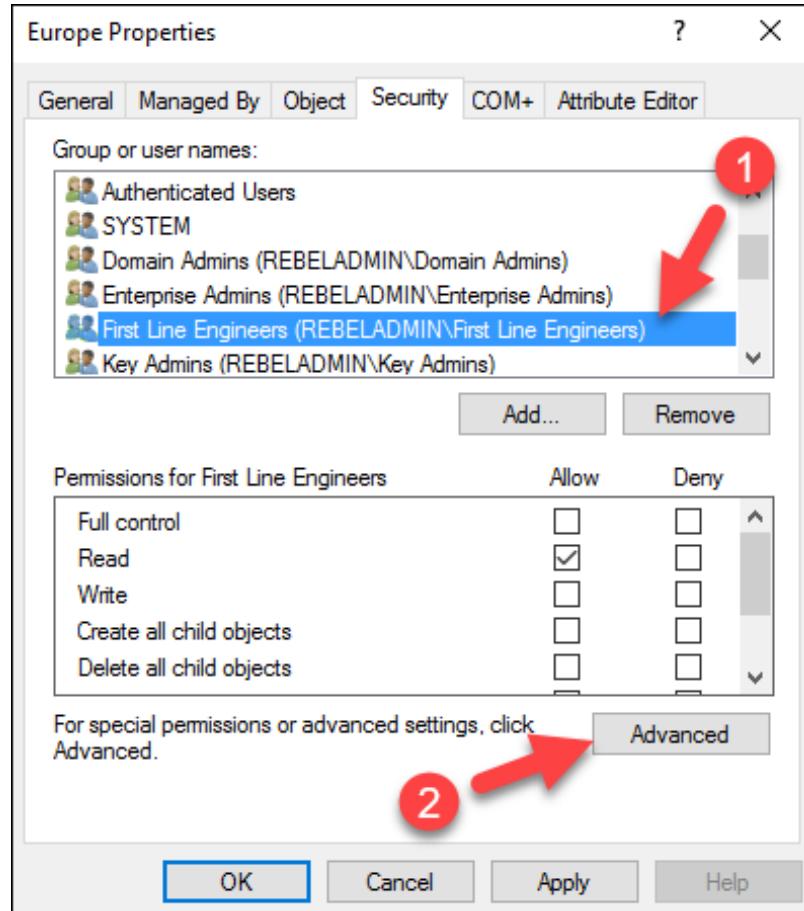


Figure 16.5: Open advanced permission settings for a group

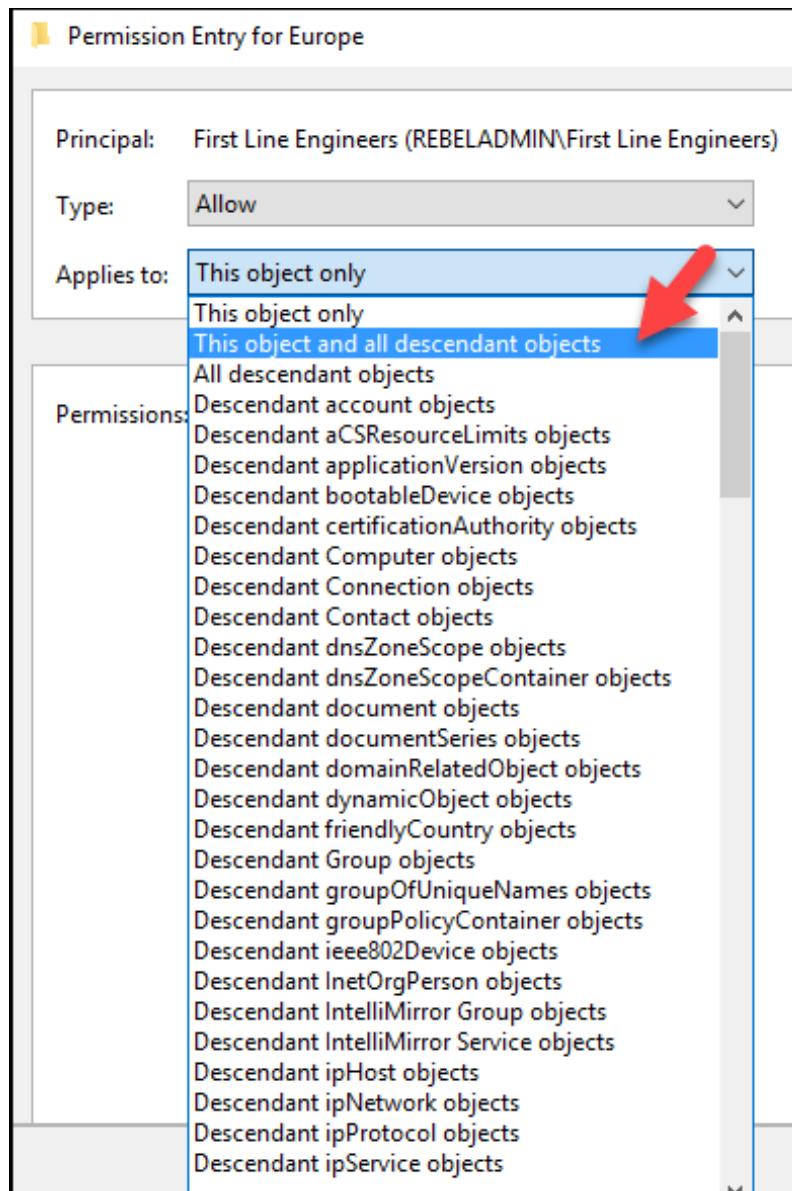


Figure 16.6: Advanced permission settings for a group

```
Administrator: PowerShell 7 (x64)
PowerShell 7.1.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Users\adam> New-ADUser -Name "Simon" -Path "OU=Users,OU=Asia,DC=rebeladmin,DC=com"
New-ADUser: Access is denied.
PS C:\Users\adam>
```

Figure 16.7: ACL permissions are preventing the provision of a new user account

```
C:\Program Files\PowerShell\7\pwsh.exe
PowerShell 7.1.4
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Users\adam> Remove-ADUser -Identity "CN=Dishan Francis,OU=Europe,DC=rebeladmin,DC=com"

Confirm
Are you sure you want to perform this action?
Performing the operation "Remove" on target "CN=Dishan Francis,OU=Europe,DC=rebeladmin,DC=com".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y
Remove-ADUser: Access is denied.
PS C:\Users\adam>
```

Figure 16.8: ACL permissions are preventing the provision of a new user account

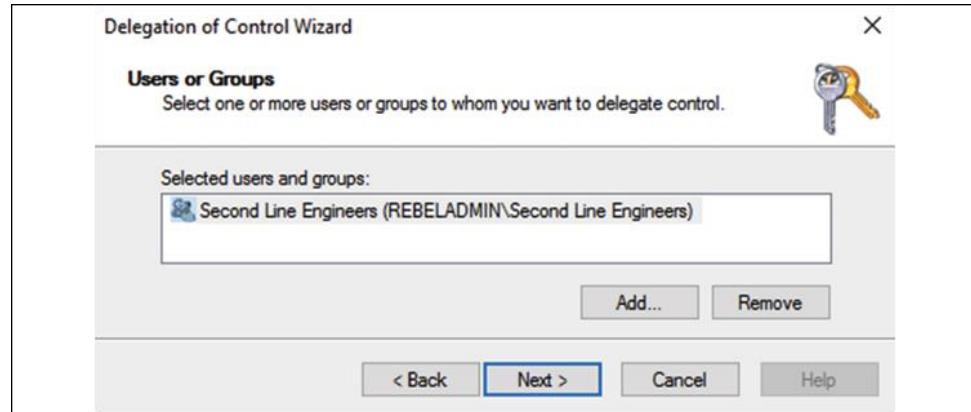


Figure 16.9: Delegating control for a group

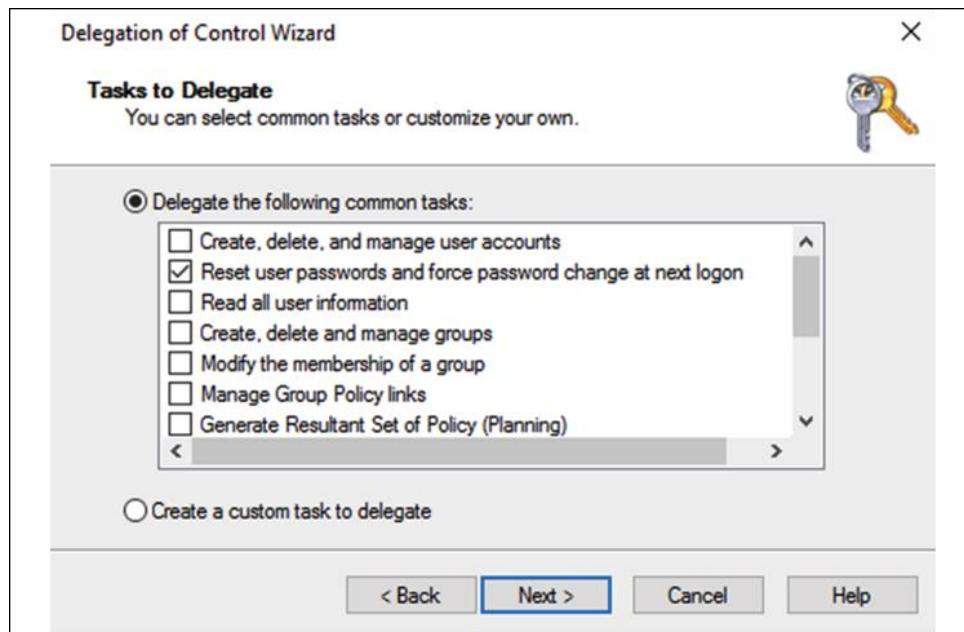


Figure 16.10: Delegate tasks

```
PS C:\Windows\System32> Remove-ADUser -Identity "CN=Dishan Francis,OU=Users,OU=Europe,DC=rebeladmin,DC=com"
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove" on target "CN=Dishan Francis,OU=Users,OU=Europe,DC=rebeladmin,DC=com".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y
Remove-ADUser: Access is denied.
PS C:\Windows\System32> _
```

Figure 16.11: Can't remove user as no delegated permissions

The screenshot shows the 'Create Password Settings' dialog for 'Help Desk Users Password Policy'. The 'Password Settings' tab is selected. It includes fields for 'Name' (Help Desk Users Password Policy), 'Precedence' (10), and various checkboxes for password complexity rules like 'Enforce minimum password length' (7 characters) and 'Enforce password history' (24 previous passwords). On the right, 'Password age options' are set with 'Enforce minimum password age' (1 day) and 'Enforce maximum password age' (42 days). Below that, 'Enforce account lockout policy' is set with 'For a duration of (mins)' (30 minutes). The 'Directly Applies To' section lists 'First Line Engineers' under 'Name'. Buttons for '?', 'X', and 'Add...' / 'Remove' are visible.

Figure 16.12: Password policy settings

```

PowerShell 7.1.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Windows\System32> Get-ADFineGrainedPasswordPolicy -Identity "Domain Admin Password Policy"

AppliesTo          : {}
ComplexityEnabled : True
DistinguishedName : CN=Domain Admin Password Policy,CN=Password Settings
                     Container,CN=System,DC=rebeladmin,DC=com
LockoutDuration   : 08:00:00
LockoutObservationWindow : 08:00:00
LockoutThreshold  : 3
MaxPasswordAge    : 30.00:00:00
MinPasswordAge    : 7.00:00:00
MinPasswordLength : 12
Name               : Domain Admin Password Policy
ObjectClass        : msDS-PasswordSettings
ObjectGUID         : bef98e70-ba1a-48d9-b762-221f22784c88
PasswordHistoryCount : 50
Precedence         : 1
ReversibleEncryptionEnabled : False

PS C:\Windows\System32> ■

```

Figure 16.13: Properties of the password policy

```

PS C:\Windows\System32> Get-ADGroup -Identity "Protected Users"

DistinguishedName : CN=Protected Users,CN=Users,DC=rebeladmin,DC=com
GroupCategory     : Security
GroupScope        : Global
Name              : Protected Users
ObjectClass       : group
ObjectGUID        : 685c8266-fdf3-4092-8207-c3a57a16f0fa
SamAccountName    : Protected Users
SID               : S-1-5-21-1495263175-677223849-3620536339-525

```

Figure 16.14: Properties of the Protected Users group

```

Authentication Id : 0x399334 (000000001002ea8d)
Session          : Interactive from 3
User Name         : liam
Domain           : REBELADMIN
Logon Server     : REBELADMIN.LOC_01
Logon Time       : 15/04/2017 08:35:20
SID               : S-1-5-21-1495263175-677223849-3620536339-525
msv :
[00001000] CredentialKeys
* NTLM : 947e1646ca1470d18fd6d976ba8d6a
* SHA1 : abc44618ab845c/ddd29ca5/f95bac318/1b6
[00001000] Preauth
* Username : liam
* Domain : REBELADMIN
* NTLM : 947e1646ca1470d18fd6d976ba8d6a
* SHA1 : abc44618ab845c/ddd29ca5/f95bac318/1b6
tspkg :
wdigest :
* Username : liam
* Domain : REBELADMIN
* Password : (null)
kerberos :
* Username : liam
* Domain : REBELADMIN.COM
* Password : (null)
ssp :
credman :

```

Figure 16.15: Viewing a hash value using Mimikatz

```

Authentication Id : 0 : 3580977 (00000000:0036a175)
Session          : Interactive - From A
User Name        : adam
Domain          : REBELADMIN
Logon Server    : REBELADMIN-01
Logon Time      : 2017-04-21T08:52:06
SID             : S-1-5-21-4041220333-1835452706-552999228-1229

msv : [B8001000] CredentialKeys
      * RootKey : Fc7b034de210b04e2b921a5811dc1165fe4a6dcfdde33b3f930d2b41981b789
      * DPAPI  : c3ebcfb1a4b912d6ef978dbd25c46

tspk8 :
  userpass :
    * Username : adam
    * Domain  : REBELADMIN
    * Password : (null)
  kerberos :
    * Username : adam
    * Domain  : REBELADMIN.COM
    * Password : (null)
  ssp :
    credman :

```

Figure 16.16: Viewing a hash value using Mimikatz of a user in the Protected Users group

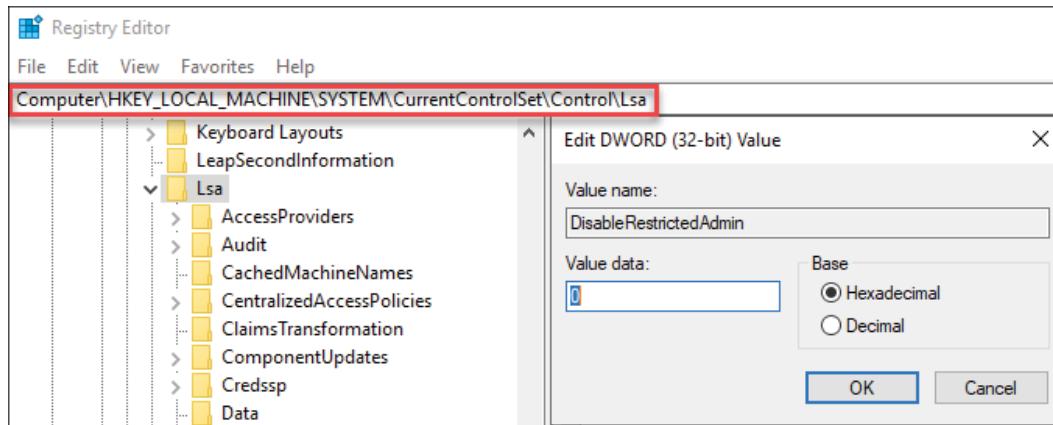


Figure 16.17: Registry key value

```

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Windows\System32> whoami
REBELADMIN\adam
PS C:\Windows\System32> whoami /groups

GROUP INFORMATION
-----
Group Name           Type      SID                                         Attributes
-----              ----      --                                         -----
Everyone            Well-known group S-1-1-0                               Mandatory group, Enabled by default, Enabled group
BUILTIN\Administrators   Alias    S-1-5-32-544                         Mandatory group, Enabled by default, Enabled group, Group owner
BUILTIN\Users         Alias    S-1-5-32-545                         Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\REMOTE INTERACTIVE          Well-known group S-1-5-14
NT AUTHORITY\SYSTEM          Well-known group S-1-5-15
NT AUTHORITY\INTERACTIVE          Well-known group S-1-5-11
NT AUTHORITY\Authenticated Users      Well-known group S-1-5-15
NT AUTHORITY\This Organization      Well-known group S-1-5-15
LOCAL                Well-known group S-1-2-0                               Mandatory group, Enabled by default, Enabled group
REBELADMIN\Domain Admins       Group    S-1-5-21-1495263175-677223840-3620536339-512 Mandatory group, Enabled by default, Enabled group
Authentication authority asserted identity Well-known group S-1-18-1                               Mandatory group, Enabled by default, Enabled group
REBELADMIN\Denies RDP Password Replication Group Alias    S-1-5-21-1495263175-677223840-3620536339-572 Mandatory group, Enabled by default, Enabled group, Local Group
Mandatory Label\High Mandatory Level Label      S-1-16-12228
PS C:\Windows\System32>

```

Figure 16.18: User's group memberships

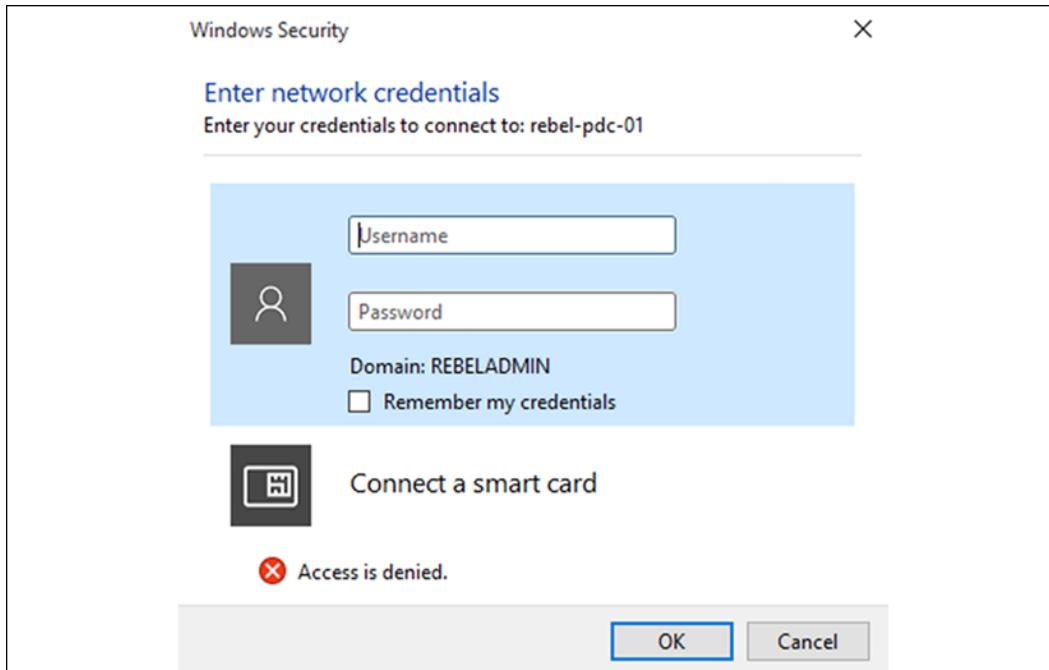


Figure 16.19: Login error

The image shows the "Server Manager" interface under the "All Servers" section. The left sidebar has "Dashboard" and "All Servers" selected. The main pane displays a table for "All servers | 1 total". The table has columns: Server Name, IPv4 Address, Manageability, Last Update, and Windows Activation. One row is shown for "REBEL-PDC-01", which is listed as "Online - Access denied 15/04/2017 23:01:21 -".

Server Name	IPv4 Address	Manageability	Last Update	Windows Activation
REBEL-PDC-01	-	Online - Access denied	15/04/2017 23:01:21	-

Figure 16.20: Server Manager access denied error

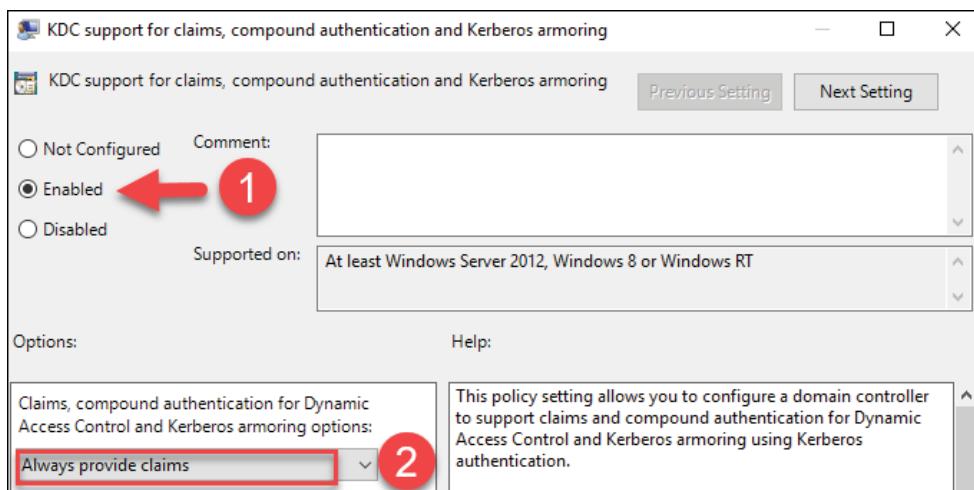


Figure 16.21: Enabling KDC support for claims, compound authentication, and Kerberos armoring policy setting

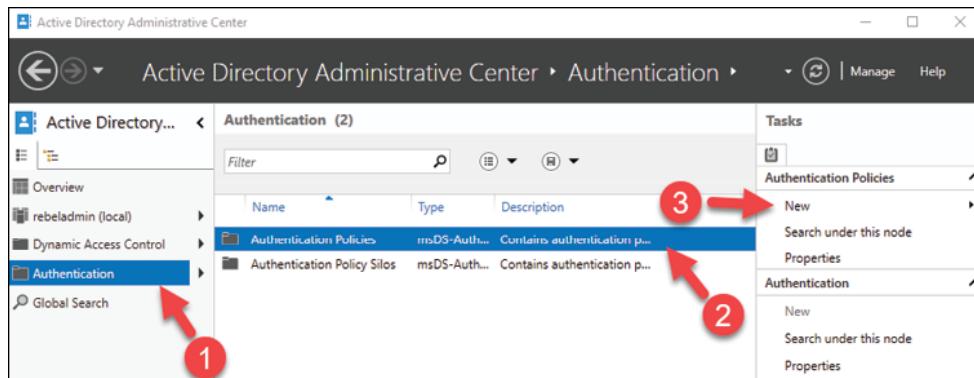


Figure 16.22: Creating a new authentication policy

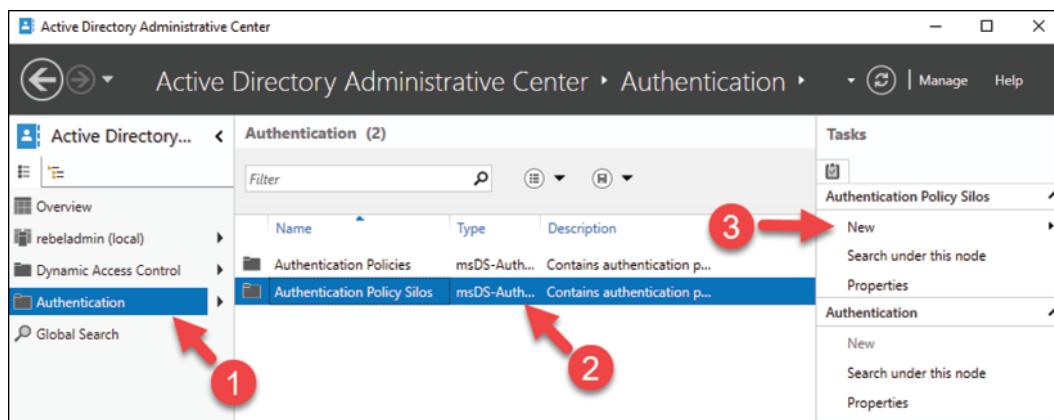


Figure 16.23: Creating new authentication policy silos

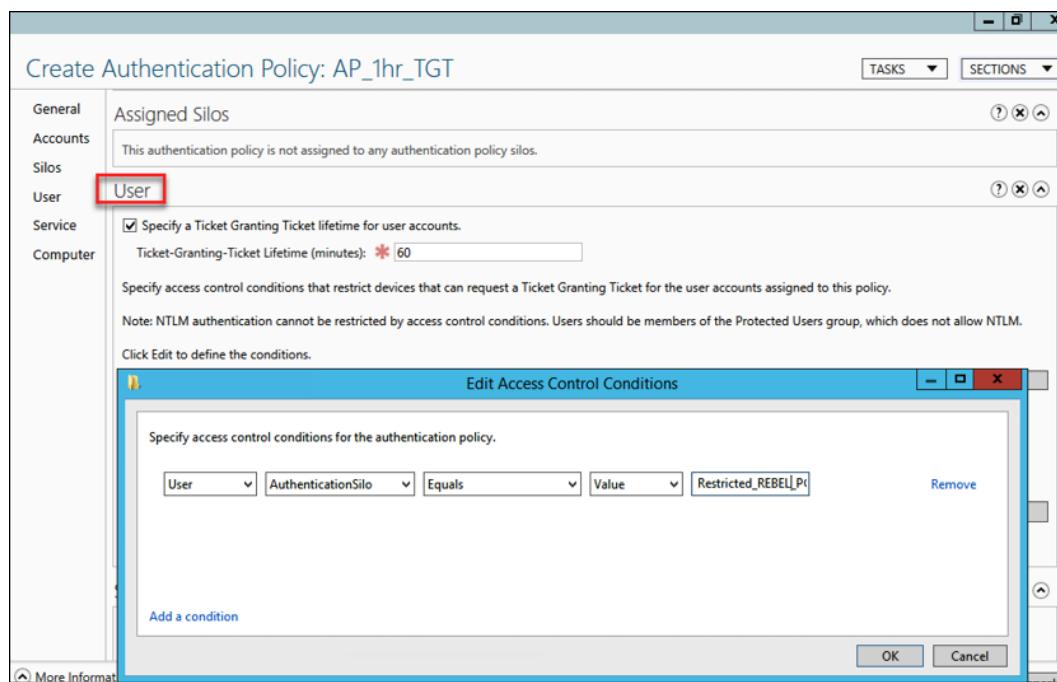


Figure 16.24: Edit Access Control Conditions

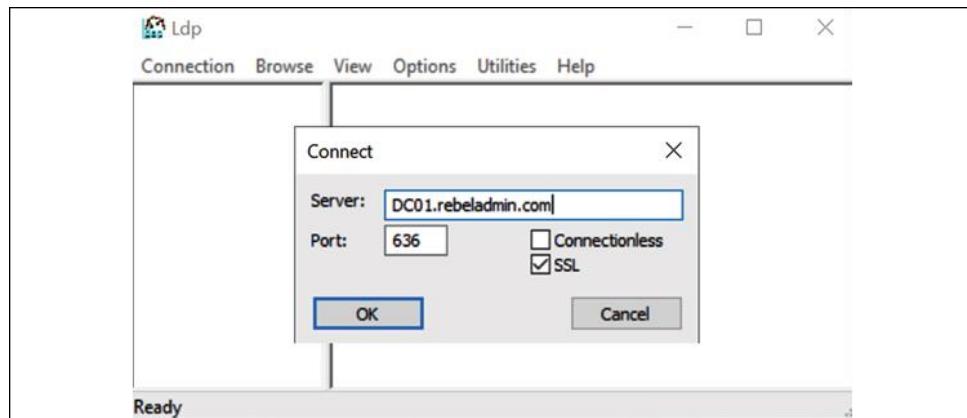


Figure 16.25: Testing a secure LDAP connection

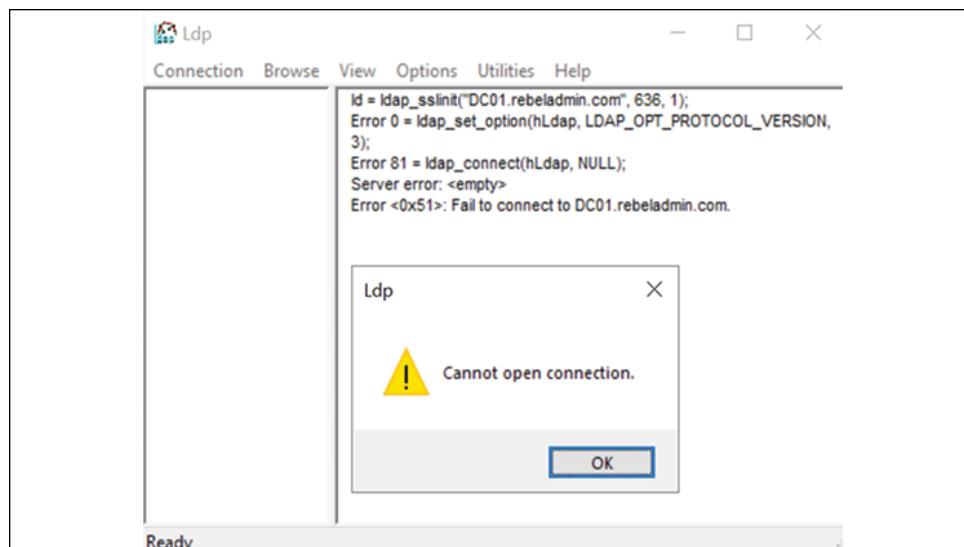


Figure 16.26: Secure LDAP connection fails

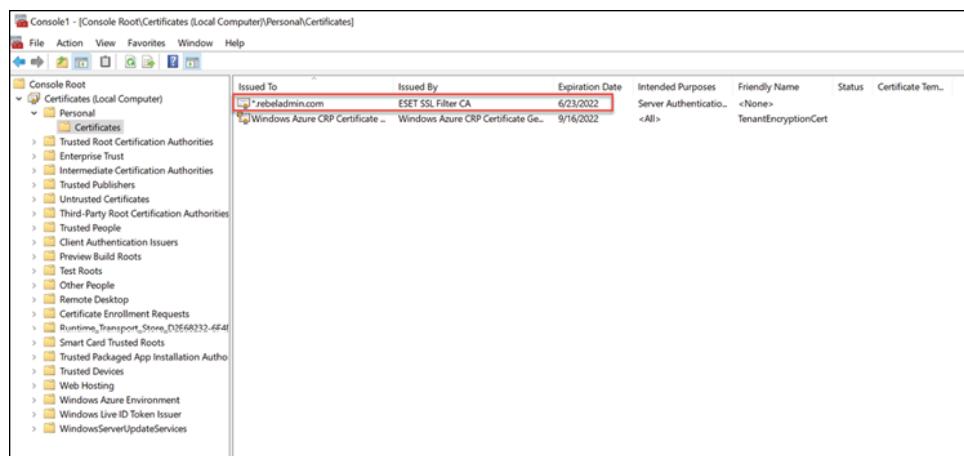


Figure 16.27: Installing a certificate in the domain controller



Figure 16.28: Successful connection with secure LDAP



Figure 16.29: Microsoft LAPS installation wizard

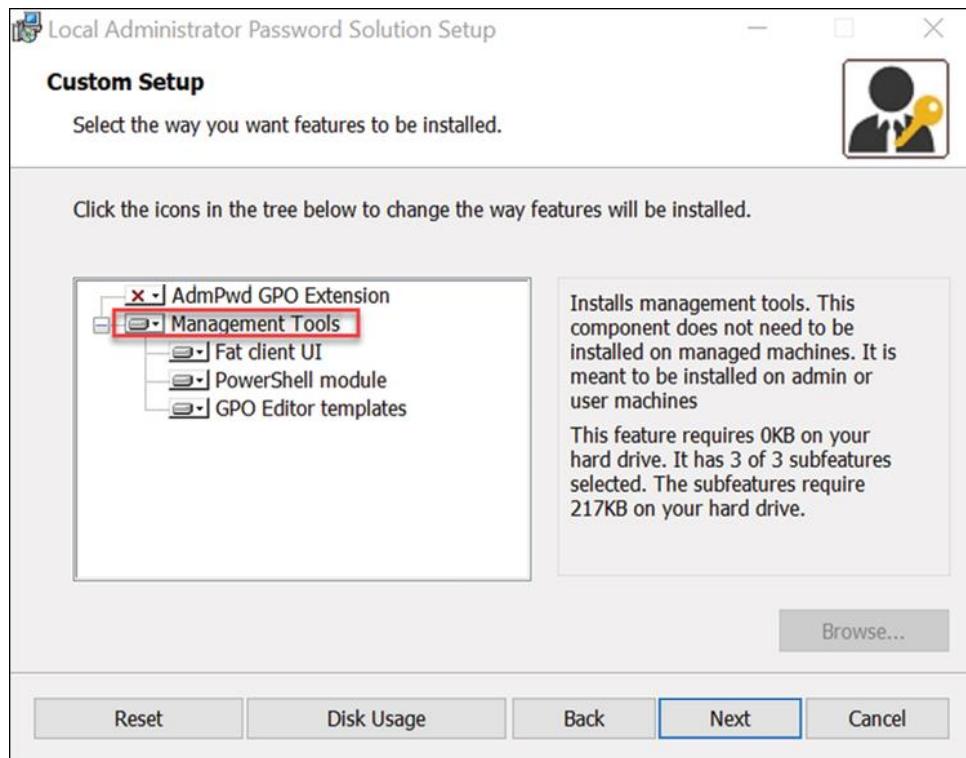


Figure 16.30: Installing Management Tools

```
PS C:\Users\dfrancis> Update-AdmPwdADSchema
Operation          DistinguishedName          Status
-----          -----
AddSchemaAttribute cn=ms-Mcs-AdmPwdExpirationTime,CN=Schema,CN=Configuration,DC=rebeladmin,DC=com Success
AddSchemaAttribute cn=ms-Mcs-AdmPwd,CN=Schema,CN=Configuration,DC=rebeladmin,DC=com Success
ModifySchemaClass  cn=computer,CN=Schema,CN=Configuration,DC=rebeladmin,DC=com Success
PS C:\Users\dfrancis>
```

Figure 16.31: Updating the AD schema

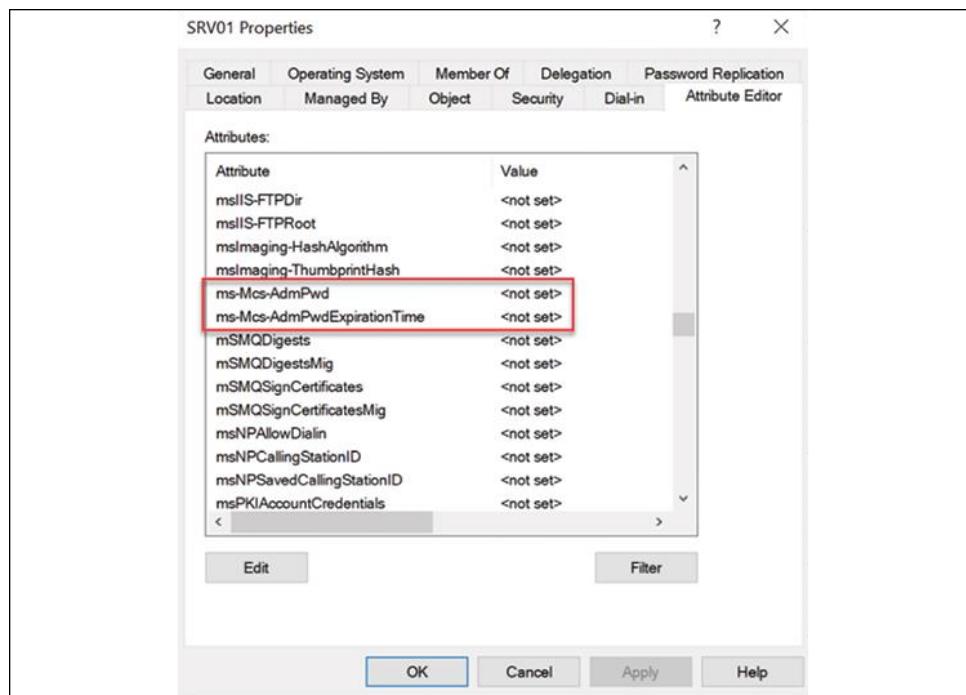


Figure 16.32: New attributes under the computer object

```
PS C:\Users\dfrancis> Set-AdmPwdComputerSelfPermission -OrgUnit RAServers
Name          DistinguishedName          Status
----          -----          -----
RAServers     OU=RAServers,DC=rebeladmin,DC=com          Delegated
PS C:\Users\dfrancis>
```

Figure 16.33: Changing computer object permissions

```
PS C:\Windows\System32> Find-AdmPwdExtendedRights -Identity "RAServers"
ObjectDN          ExtendedRightHolders
-----          -----
OU=RAServers,DC=rebeladmin,DC=com          {NT AUTHORITY\SYSTEM, REBELADMIN\Domain Admins}
PS C:\Windows\System32>
```

Figure 16.34: Verifying extended rights

```

PS C:\Windows\System32> Find-AdmPwdExtendedRights -Identity "RAServers" | fl
RunspaceId          : e7e145eb-a254-457a-a987-b93d3dfe28a8
ObjectDN            : OU=RAServers,DC=rebeladmin,DC=com
ExtendedRightHolders : {NT AUTHORITY\SYSTEM, REBELADMIN\Domain Admins, REBELADMIN\ITAdmins}

PS C:\Windows\System32>

```

Figure 16.35: Updated extended rights



Figure 16.36: Installing an agent using a GPO

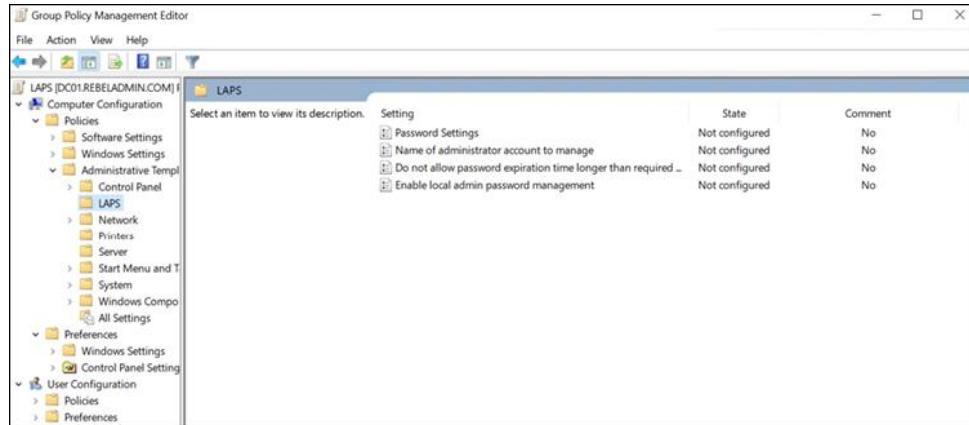


Figure 16.37: Microsoft LAPS GPO settings

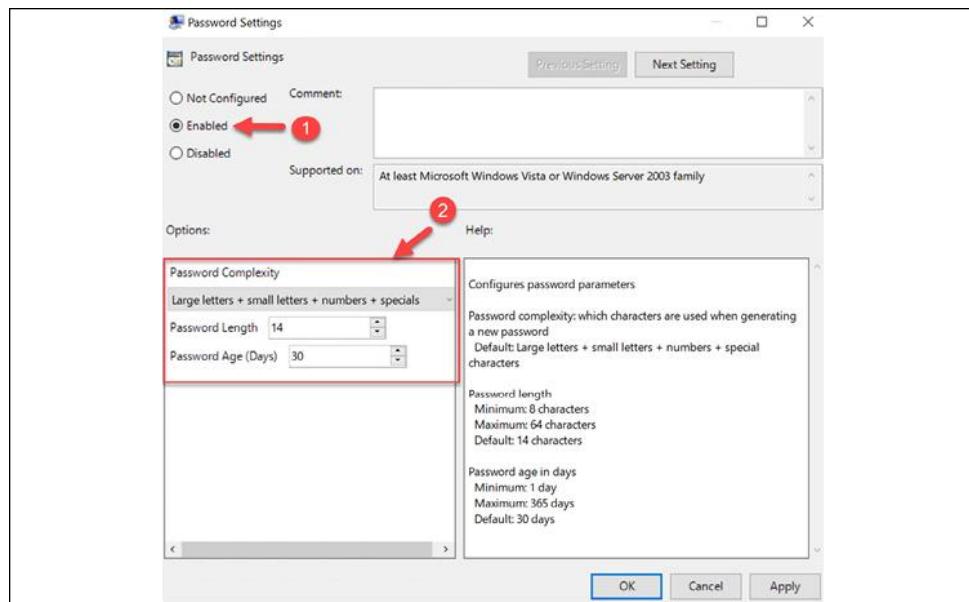


Figure 16.38: Microsoft LAPS GPO settings (password)

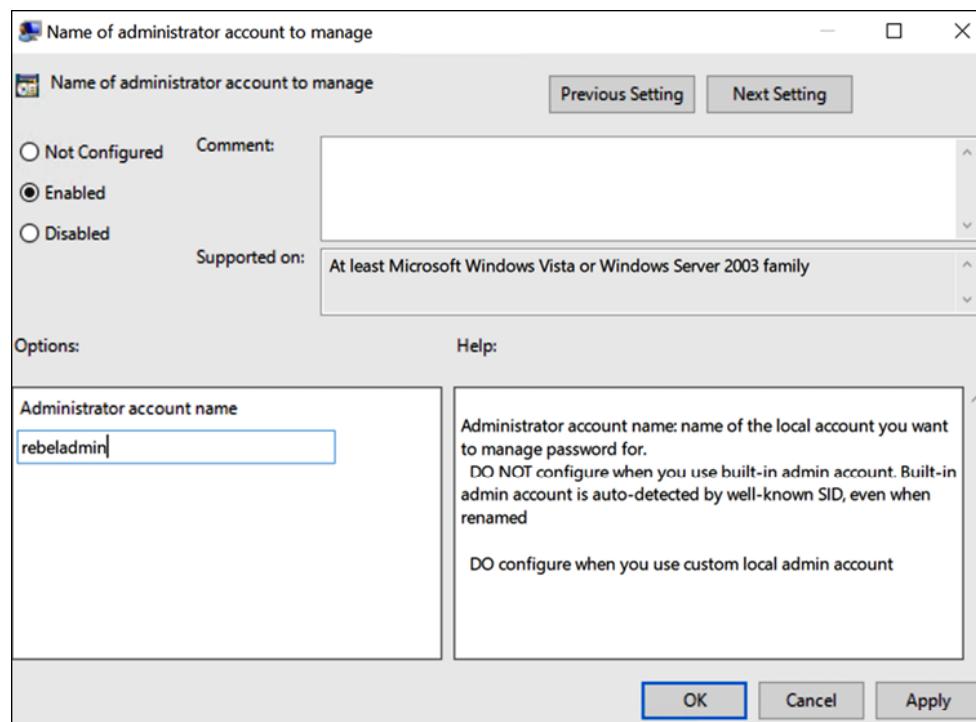


Figure 16.39: Microsoft LAPS GPO settings (Name of administrator account to manage)

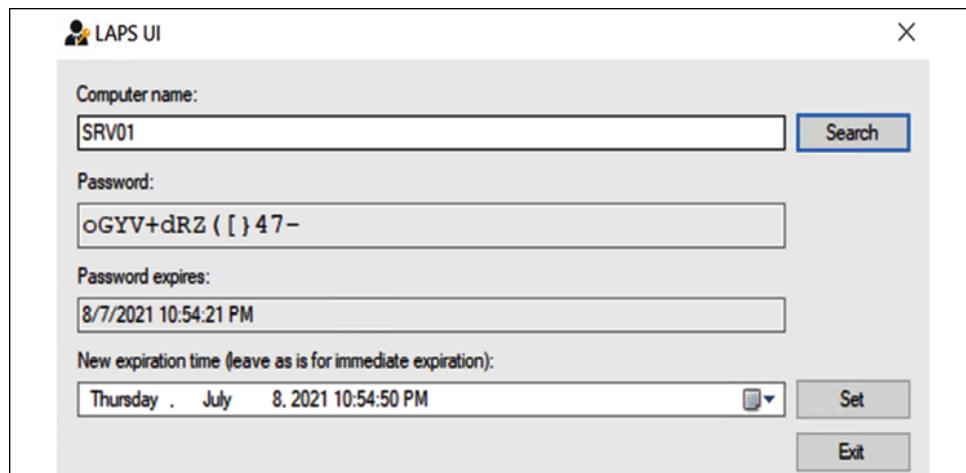


Figure 16.40: Checking the local administrator password using LAPS UI

PS C:\Windows\System32> Get-AdmPwdPassword -ComputerName SRV01			
ComputerName	DistinguishedName	Password	ExpirationTimestamp
SRV01	CN=SRV01,OU=RAServers,DC=rebeladmin,DC=com	oGYV+dRZ([]47 -)	8/7/2021 10:54:21 PM

Figure 16.41: Checking the local administrator password using PowerShell

The screenshot shows the Azure AD Security blade under Authentication methods. The Password protection section is selected. The configuration includes:

- Custom smart lockout: Lockout threshold (10), Lockout duration in seconds (60).
- Custom banned passwords: Enforce custom list (Yes), Custom banned password list (rebeladmin).
- Password protection for Windows Server Active Directory: Enable password protection on Windows Server Active Directory (Yes).
- Mode: Enforced.

Figure 16.42: Enabling password protection on Windows AD

```
PS C:\Users\dfrancis.M365X420225> Set-ADAccountPassword -Identity testuser -Reset -NewPassword (ConvertTo-SecureString -AsPlainText "rebeladmin@A123" -Force)
Set-ADAccountPassword: The password does not meet the length, complexity, or history requirement of the domain.
```

Figure 16.43: Test user password change

Commands

Command 16.1

```
New-ADUser -Name "Dale"  
-Path "OU=Users,OU=Europe,DC=rebeladmin,DC=com"
```

Command 16.2

```
New-ADUser -Name "Simon"  
-Path "OU=Users,OU=Asia,DC=rebeladmin,DC=com"
```

Command 16.3

```
Remove-ADUser -Identity "CN=Dishan Francis,  
OU=Users,OU=Europe,DC=rebeladmin,DC=com"
```

Command 16.4

```
Set-ADAccountPassword -Identity difrancis
```

Command 16.5

```
Remove-ADUser -Identity "CN=Dishan Francis,  
OU=Users,OU=Europe,DC=rebeladmin,DC=com"16.6
```

```
New-ADFineGrainedPasswordPolicy -Name "Domain Admin Password Policy" -  
Precedence 1  
-MinPasswordLength 12 -MaxPasswordAge "30" -MinPasswordAge "7" -  
-PasswordHistoryCount 50 -ComplexityEnabled:$true -  
-LockoutDuration "8:00" -  
-LockoutObservationWindow "8:00" -LockoutThreshold 3 -  
-ReversibleEncryptionEnabled:$false
```

Command 16.7

```
Get-ADFineGrainedPasswordPolicy -Identity "Domain Admin Password  
Policy"
```

Command 16.8

```
Add-ADFineGrainedPasswordPolicySubject -Identity "Domain Admin  
Password Policy" -Subjects "Domain Admins"16.9
```

```
Get-ADFineGrainedPasswordPolicy -Identity "Domain Admin Password Policy" | Format-Table AppliesTo -AutoSize
```

Command 16.10

```
Get-ADFineGrainedPasswordPolicy -Filter * | Format-Table Name,Precedence,AppliesTo -AutoSize
```

Command 16.11

```
Get-ADGroup -Identity "Protected Users"
```

Command 16.12

```
Get-ADGroup -Identity "Protected Users" | Add-ADGroupMember -Members "CN=Adam,CN=Users,DC=rebeladmin,DC=com"
```

Command 16.13

```
Get-ADGroupMember -Identity "Protected Users"
```

Command 16.14

```
New-ADAAuthenticationPolicy -Name "AP_1hr_TGT" -UserTGTLifetimeMins 60 -Enforce
```

Command 16.15

```
New-ADAAuthenticationPolicySilo -Name Restricted_REBEL_PC01 -UserAuthenticationPolicy AP_1hr_TGT -ComputerAuthenticationPolicy AP_1hr_TGT -ServiceAuthenticationPolicy AP_1hr_TGT -Enforce
```

Command 16.16

```
Grant-ADAAuthenticationPolicySiloAccess -Identity Restricted_REBEL_PC01 -Account Peter
```

Command 16.17

```
Get-ADComputer -Filter 'Name -like "REBEL-PC01"' | Grant-ADAAuthenticationPolicySiloAccess -Identity Restricted_REBEL_PC01
```

Command 16.18

```
Set-ADAccountAuthenticationPolicySilo -Identity Peter -  
AuthenticationPolicySilo Restricted_REBEL_PC01 -AuthenticationPolicy  
AP_1hr_TGT
```

Command 16.19

```
Get-ADComputer -Filter 'Name -like "REBEL-PC01"' | Set-  
ADAccountAuthenticationPolicySilo -AuthenticationPolicySilo  
Restricted_REBEL_PC01 -AuthenticationPolicy AP_1hr_TGT
```

Command 16.20

```
Set-ADAAuthenticationPolicy -Identity AP_1hr_TGT -  
UserAllowedToAuthenticateFrom  
"O:SYG:SYD:(XA;OICI;CR;;WD:(@USER.ad://ext/AuthenticationSilo ==  
`"Restricted_REBEL_PC01`"))"
```

Command 16.21

```
Import-module AdmPwd.PS
```

Command 16.22

```
Set-AdmPwdComputerSelfPermission -OrgUnit RAServers
```

Command 16.23

```
Import-module AdmPwd.PS
```

Command 16.24

```
Find-AdmPwdExtendedRights -Identity "RAServers"
```

Command 16.25

```
Set-AdmPwdReadPasswordPermission -Identity "RAServers" -  
AllowedPrincipals "ITAdmins"
```

Command 16.26

```
Find-AdmPwdExtendedRights -Identity "RAServers" | fl
```

Command 16.27

```
Get-AdmPwdPassword -ComputerName SRV01
```

Command 16.28

```
Import-Module AzureADPasswordProtection  
Register-AzureADPasswordProtectionProxy -AccountUpn  
'admin@rebeladm.onmicrosoft.com'
```

Command 16.29

```
Import-Module AzureADPasswordProtection  
Register-AzureADPasswordProtectionForest -AccountUpn  
'admin@rebeladm.onmicrosoft.com'
```

Command 16.30

```
Set-ADAccountPassword -Identity testuser -Reset -NewPassword  
(ConvertTo-SecureString -AsPlainText "rebeladmin@A123" -Force)
```

Chapter 17

Advanced AD Management with PowerShell

The very first **Active Directory (AD)** instance I set up was based on Windows Server 2003. It was a completely different approach from today's Active Directory installations. In Windows Server 2003, there were a lot of prerequisite tasks, such as installing a DNS role, setting up DNS zones, and adding the domain prefix. Even those tasks were directly related to **Active Directory Domain Services (AD DS)**, and I had to configure them separately prior to running the `DCPROMO.exe` command. But today, the Active Directory role installation process is very straightforward. With basic knowledge and resources, anyone can get a domain controller up and running with a few clicks.

Microsoft has made server role installations and configurations easy over the years, not just AD DS. The main reason behind all these enhancements was to save time for engineers. Installations, configurations, and repetitive infrastructure tasks take up the majority of an engineer's time. Also with the pandemic, an engineer has to wear many hats as businesses shrink IT budgets. To save time on repetitive administrative tasks, we should be looking at automation technologies. In the early days, we used `DOS` commands, VBScript, and batch files to automate administrative tasks. But there were problems with that. Applications, server roles, and services had limitations on working with these automation technologies. Not every function available in the GUI supported the use of commands or scripts. This lack of support and lack of flexibility was holding engineers back from automating tasks.

To bring automation to the next level, Microsoft released a more flexible, more powerful, more integrated scripting language.

PowerShell 1.0 was the starting point and it was available to the public from November 2006. During the last decade, there have been a few versions released and it's now at version 5.1 (mainstream). Microsoft also released a separate version of PowerShell called PowerShell Core 6.0 (January 10, 2018). It was compatible with Linux and macOS as well. Now it has been replaced by PowerShell 7.0. In *Chapter 2, Active Directory Domain Services 2022*, I mentioned PowerShell 7 and throughout this book, I have used PowerShell 7 for configurations and the administration of Active Directory roles. In this chapter, I will explain how we can use PowerShell to further improve AD DS environment management.

We are also going to look at managing identities in a hybrid environment using Azure AD PowerShell and Microsoft Graph.

The cmdlets and scripts used in this chapter were written and tested in an environment that has the following:

- Windows Server 2022
- An AD domain and forest functional level set to Windows Server 2016
- PowerShell 7.1

- Azure AD Premium P2

In this chapter, we will cover the following topics:

- PowerShell scripts and commands that can be used to manage AD objects
- PowerShell scripts and commands that can be used to manage and troubleshoot AD replication
- PowerShell scripts and commands that can be used to manage identities in a hybrid environment using the Azure AD PowerShell module
- Microsoft Graph for Hybrid Identity management

Before we start using PowerShell for Active Directory management, we need to make sure relevant tools are installed and configured.

AD management with PowerShell – preparation

A PowerShell module includes assemblies, scripts, and functionalities. In order to use the functionalities, we need to import the module. After that, we can call for the contents of the module to manage relevant server roles, services, or features.

Before we start Active Directory management with PowerShell, first we need to import the ActiveDirectory module.

There are a few ways to do this. These include installing the AD DS server role or by installing **Remote Server Administration Tools (RSAT)**:

- **AD DS server role:**
 - a. If we install the AD DS server role using Server Manager, the Active Directory module for Windows PowerShell is installed as a feature:

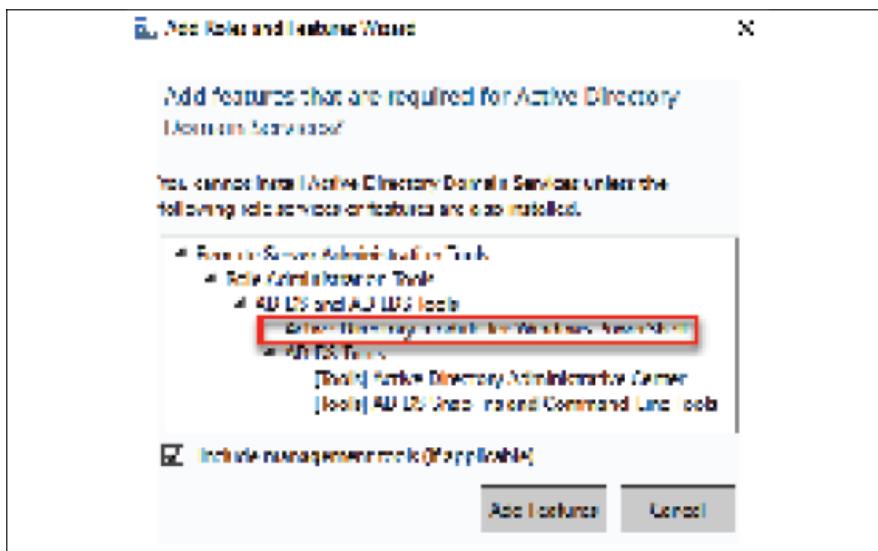


Figure 17.1: Active Directory module for Windows PowerShell feature

- b. If the AD DS role is installed using PowerShell, we need to include the management tools by using -IncludeManagementTools. Otherwise, by default, it will not install the module:

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

- **Remote Server Administration Tools:**

- c. Even if the server doesn't have the AD DS role installed, the existing domain environment can be managed using the AD DS PowerShell module. The AD PowerShell module is included with RSAT and can be installed using Server Manager or PowerShell.
- d. On Server Manager, it can be found by navigating to **Features | Remote Server Administration Tools | Role Administration Tools | AD DS and AD LDS Tools | Active Directory module for PowerShell**, as shown in the following screenshot:

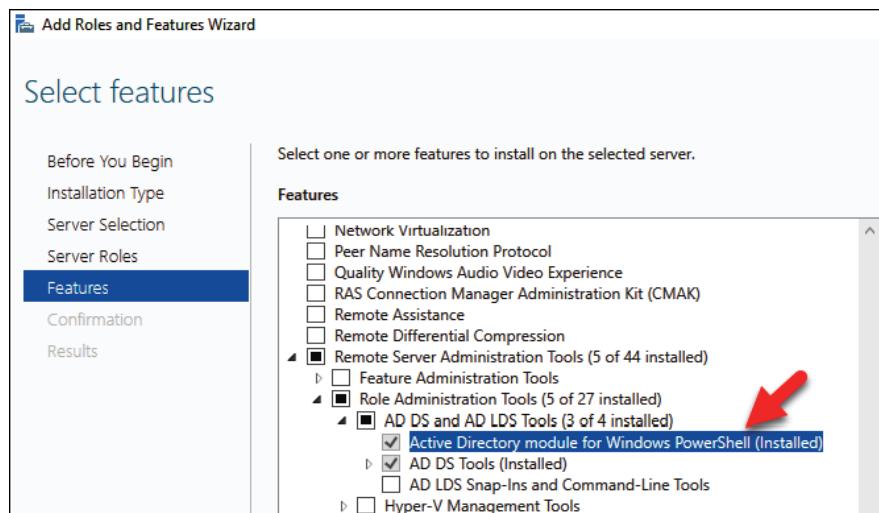


Figure 17.2: Active Directory module for Windows PowerShell feature under RSAT

- e. It can also be installed using PowerShell:

```
Add-WindowsFeature RSAT-AD-PowerShell
```

It is also possible to install RSAT on the Windows desktop OS. As an example, RSAT for Windows 10 can be downloaded from <https://bit.ly/3HZ5k0W>.

PowerShell 7

In the preceding section, I explained how we can prepare native Windows PowerShell for Active Directory administration. Most of the scripts and commands used in this book also work with native Windows PowerShell. But PowerShell 7 works in a different way. It doesn't come as part of Windows and needs to be installed separately as an application. To install PowerShell 7, please follow the following guide: <https://bit.ly/30Pu4HM>

After prerequisites are in place, we can list all the commands available under the module using the following command:

```
Get-Command -Module ActiveDirectory
```

There are about 147 commands under the module. The complete syntax for any command can be viewed using this command:

```
Get-Command commandname -Syntax
```

As an example, the following command will list the syntax for the New-ADUser command:

```
Get-Command New-ADUser -Syntax
```

The `Get-Help` command provides help for any command. As an example, the following command provides help for the `New-ADUser` command:

```
Get-Help New-ADUser
```

We also can view an example for the New-ADUser command using this:

```
Get-Help New-ADUser -Example
```

More information on the command can be viewed using this:

```
Get-Help New-ADUser -Detailed
```

Technical information on the command can be viewed using the following:

```
Get-Help New-ADUser -Full
```

Online information about the command can be viewed using this:

```
Get-Help New-ADUser -Online
```

In this section, we learned how to install the Active Directory module for PowerShell. We also learned about the basic functions of the module. Now, let's move on and further explore the Active Directory management capabilities of the module.

AD management commands and scripts

The module has 147 commands, and they can be used in countless different ways to manage the Active Directory environment. In this section, we will look at the capabilities of these commands and see how we can use them to improve Active Directory management.

I'd like to start this section by explaining how we can review the existing configuration of an Active Directory environment. The quick way to review the directory server configuration and capabilities is to use the following command:

```
Get-ADRootDSE
```

This command provides important information, such as forest and domain functional levels, the default naming context, the current time, and the currently logged-in domain controller.

The next step is to find the domain controllers in the domain. We can use the following to list the domain controller name, the IP address, the status of the global catalog server, and the **Flexible Single Master Operation (FSMO)** roles:

```
Get-ADDomainController -Filter * | Select-Object  
Name,IPv4Address,IsGlobalCatalog,OperationMasterRoles
```

It is also important to know about the Active Directory site as it explains the physical topology of Active Directory:

```
Get-ADDomainController -Filter * | Select-Object Name,IPv4Address,Site
```

An Active Directory forest can have multiple domains. The following commands will list the forest names, the domain name, the domain controller, the IP address, and the Active Directory site:

```
$Forestwide = (Get-ADForest).Domains | % { Get-ADDomainController -  
Filter * -Server $_ }  
write-output $Forestwide -Filter * | Select-Object  
Name,Forest,Domain,IPv4Address,Site
```

If we know the domain name, we can list the domain controllers and the **read-only domain controller (RODC)** using the following command:

```
$Domain = Read-Host 'What is your Domain Name ?'  
Get-ADDomain -Identity $Domain | select  
ReplicaDirectoryServers,ReadOnlyReplicaDirectoryServer
```

With this command, the system will ask the user to input the domain name. Once the user replies, it lists the domain controllers.

In the preceding command, ReplicaDirectoryServers represents the read and write domain controllers, and ReadOnlyReplicaDirectoryServer represents the RODCs.

Replication

Data replication is crucial for a healthy Active Directory environment. For a given domain controller, we can find its inbound replication partners using this:

```
Get-ADReplicationPartnerMetadata -Target REBEL-SRV01.rebeladmin.com
```

The preceding command provides a detailed description of the replication health of the given domain controller, including the last successful replication, replication partition, server, and so on.

We can list all the inbound replication partners for the given domain using the following command:

```
Get-ADReplicationPartnerMetadata -Target "rebeladmin.com" -Scope  
Domain
```

In the preceding command, the scope is defined as the domain. This can be changed to the forest to get a list of the inbound partners in the forest. The output is based on the default partition. If needed, the

partition can be changed using `-Partition` to a configuration or schema partition. It will list the relevant inbound partners for the selected partition.

The associated replication failures for a site, forest, domain, and domain controller can be found using the `Get-ADReplicationFailure` cmdlet:

```
Get-ADReplicationFailure -Target REBEL-SRV01.rebeladmin.com
```

The preceding command will list the replication failures for the given domain controller.

Replication failures for the domain can be found using this:

```
Get-ADReplicationFailure -Target rebeladmin.com -Scope Domain
```

Replication failures for the forest can be found using the following command:

```
Get-ADReplicationFailure -Target rebeladmin.com -Scope Forest
```

Replication failures for the site can be found using the following command:

```
Get-ADReplicationFailure -Target LondonSite -Scope Site
```

In the preceding command, `LondonSite` can be replaced with a relevant site name.

Using both `Get-ADReplicationPartnerMetadata` and `Get-ADReplicationFailure`, I have created the following PowerShell script to generate a replication health report against a specific domain controller.

The first part of the script is used to define the objects that we'll use throughout the script:

```
## Active Directory Domain Controller Replication Status##
$domaincontroller = Read-Host 'What is your Domain Controller?'
## Define Objects ##
$report = New-Object PSObject -Property @{
    ReplicationPartners = $null
    LastReplication = $null
    FailureCount = $null
    FailureType = $null
    FirstFailure = $null
}
```

In the preceding script, I have given an option for the engineer to specify the name of the domain controller:

```
$domaincontroller = Read-Host 'What is your Domain Controller?'
```

In the next part of the script, I am collecting the following data, which describes the replication connection status with the other domain controllers:

- Replication partner (`ReplicationPartners`)
- Last successful replication (`LastReplication`)

```
## Replication Partners ##
$report.ReplicationPartners = (Get-ADReplicationPartnerMetadata -Target $domaincontroller).Partner
$report.LastReplication = (Get-ADReplicationPartnerMetadata -Target $domaincontroller).LastReplicationSuccess
```

Then, I also gather the following data, which helps engineers to troubleshoot replication issues, if any exist:

- Active Directory replication failure count (FailureCount)
- Active Directory replication failure type (FailureType)
- Active Directory replication failure first recorded time (FirstFailure)

```
## Replication Failures ~##
$report.FailureCount = (Get-ADReplicationFailure -Target $domaincontroller).FailureCount
$report.FailureType = (Get-ADReplicationFailure -Target $domaincontroller).FailureType
$report.FirstFailure = (Get-ADReplicationFailure -Target $domaincontroller).FirstFailureTime
```

The last part of the script formats the output of the collected data:

```
## Format Output ##
$report | select
ReplicationPartners,LastReplication,FirstFailure,FailureCount,FailureType | Out-GridView
```

The aforementioned script is displayed in an easy way for readers to understand. When it is used in PowerShell, make sure to prevent extra line spaces.

Further to Active Directory replication topologies, there are two types of replication:

- **Intra-site:** Replication between domain controllers in the same Active Directory site
- **Inter-site:** Replication between domain controllers in different Active Directory sites

We can review AD replication site objects using the Get-ADReplicationSite cmdlet. The following command returns all the Active Directory replication sites in the Active Directory forest:

```
Get-ADReplicationSite -Filter *
```

We can review Active Directory replication site links on the Active Directory forest using the following command:

```
Get-ADReplicationSiteLink -Filter *
```

In site links, the most important information is to know the site cost and the replication schedule. This allows us to understand the replication topology and expected delays in replication.

The following command lists all the replication site links, which includes the CanadaSite along with the site link name, link cost, and replication frequency:

```
Get-ADReplicationSiteLink -Filter {SitesIncluded -eq "CanadaSite"} | Format-Table Name,Cost,ReplicationFrequencyInMinutes -AutoSize
```

A site link bridge can be used to bundle two or more site links and enable transitivity between site links.

Site link bridge information can be retrieved using the following command:

```
Get-ADReplicationSiteLinkBridge -Filter *
```

An AD site uses multiple IP subnets that are assigned to sites for its operations. It is important to associate these subnets with AD sites so that domain controllers know which computer is located at which site.

The following command will list all the subnets in the forest in a table with the subnet name and Active Directory site:

```
Get-ADReplicationSubnet -Filter * | Format-Table Name,Site -AutoSize
```

Bridgehead servers operate as the primary communication point to handle the replication data that comes in and goes out of the Active Directory site.

We can list all the preferred bridgehead servers in a domain:

```
$BHservers = ([adsi]"LDAP://CN=IP,CN=Inter-Site Transports,CN=Sites,CN=Configuration,DC=rebeladmin,DC=com").bridgeheadServerListBL  
$BHservers | Out-GridView
```

In the preceding command, the `bridgeheadServerListBL` attribute value is retrieved via the ADSI connection.

Information about the replication topology helps engineers in many ways, especially if engineers are troubleshooting Active Directory replication issues or performing an Active Directory audit. By using the preceding commands, I have created the following script to gather Active Directory replication topology data in one go.

As usual, the first part of the script is dedicated to defining objects:

```
## Script to gather information about Replication Topology ##  
## Define Objects ##  
$replreport = New-Object PSObject -Property @{  
Domain = $null  
}
```

Before we move on to the replication, it is good to collect the Active Directory domain information. This is important if an organization is using multiple domains as we can easily separate the reports:

```
## Find Domain Information ##  
$replreport.Domain = (Get-ADDomain).DNSroot
```

I have used the next section of the script to list the Active Directory sites:

```
## List down the AD sites in the Domain ##
$a = (Get-ADReplicationSite -Filter *)
Write-Host "#####" $replreport.Domain "Domain AD Sites" #####
$a | Format-Table Description,Name -AutoSize
```

Then, I am going to collect data about the Active Directory replication site link and the Active Directory replication site link bridge by using the following:

```
## List down Replication Site link Information ##
$b = (Get-ADReplicationSiteLink -Filter *)
Write-Host "#####" $replreport.Domain "Domain AD Replication
SiteLink Information" #####
$b | Format-Table Name,Cost,ReplicationFrequencyInMinutes -AutoSize
## List down SiteLink Bridge Information ##
$c = (Get-ADReplicationSiteLinkBridge -Filter *)
Write-Host "#####" $replreport.Domain "Domain AD SiteLink Bridge
Information" #####
$c | select Name,SiteLinksIncluded | Format-List
```

In a computer network, there can be multiple IP subnets. These subnets need to be assigned correctly to Active Directory sites. This way, Active Directory Domain Controller computers know which site they belong to. This also has a direct impact on Active Directory replication. In the next section, we are going to collect Active Directory subnet information and the preferred bridgehead servers for the domain:

```
## List down Subnet Information ##
$d = (Get-ADReplicationSubnet -Filter * | select Name,Site)
Write-Host "#####" $replreport.Domain "Domain Subnet Information"
#####
$d | Format-Table Name,Site -AutoSize
## List down Preferred BridgeHead Servers ##
$e = ([adsi]"LDAP://CN=IP,CN=Inter-Site
Transports,CN=Sites,CN=Configuration,DC=rebeladmin,DC=com").bridgehead
ServerListBL
Write-Host "#####" $replreport.Domain "Domain Preferred BridgeHead
Servers" #####
$e
## End of the Script ##
```

The aforementioned script is displayed in a way that's easy for readers to understand. When it is used in PowerShell, make sure to prevent extra line spaces.

In the preceding script, we need to replace the ADSI connection with the relevant domain name:

```
$e = ([adsi]"LDAP://CN=IP,CN=Inter-Site
Transports,CN=Sites,CN=Configuration,DC=rebeladmin,DC=com")
```

Healthy replication is critical for Active Directory Domain controllers. The time it takes to replicate a change to all the domain controllers depends on the number of domain controllers in place,

geographical locations, replication schedules, etc. In some situations, we have to force the replication of objects and in the next section, we are going to look into it in detail.

Replicating a specific object

Once an object is added to a domain controller, it needs to be replicated to all other domain controllers. Otherwise, users will face issues during login using AD-integrated applications and services. The replication is dependent on many different factors, such as the replication schedule and intra-site connectivity. Sometimes, we need to force the replication between domain controllers:

```
## Replicate Object From Domain Controller to Another ##
$myobject = Read-Host 'What is your AD Object Includes ?'
$sourcedc = Read-Host 'What is the Source DC ?'
$destinationdc = Read-Host 'What is the Destination DC ?'
$passobject = (Get-ADObject -Filter {Name -Like $myobject})
Sync-ADObject -object $passobject -source $sourcedc -destination
$destinationdc
Write-Host "Given Object Replicated to" $destinationdc
```

The preceding script will ask a few questions:

- **Name of object:** This need not be a **distinguished name (DN)**. All that is needed is that text be included in the object name field.
- **Source DC:** The hostname of the source DC.
- **Destination DC:** The hostname of the destination DC.

Once the relevant information is provided, the object will be forcibly replicated:

```
PS C:\Windows\system32> ## Replicate Objects to Domain Controllers ##
$myobject = Read-Host 'What is your AD Object Includes ?'
$sourcedc = Read-Host 'What is the Source DC ?'
$destinationdc = Read-Host 'What is the Destination DC ?'
$passobject = (Get-ADObject -Filter {Name -Like $myobject})
Sync-ADObject -object $passobject -source $sourcedc -destination $destinationdc
Write-Host "Given Object Replicated to" $destinationdc
What is your AD Object Includes ?: Adam
What is the Source DC ?: REBEL-PDC-01
What is the Destination DC ?: REBEL-SRV01
Given Object Replicated to REBEL-SRV01

PS C:\Windows\system32>
```

Figure 17.3: Replicating a specific object

In this section of the chapter, we learned how the Active Directory module for PowerShell can be used to review the topology of an Active Directory environment. We also learned how we can audit, troubleshoot, and manage Active Directory replication using PowerShell. In the next section, we are going to look into Active Directory object management.

Users and groups

In this section, let's look at PowerShell commands and scripts that we can use to manage AD users and groups.

Last logon time

On certain occasions, we are required to find when a user successfully logs on to a domain. This can be for audit purposes or for troubleshooting purposes:

```
$username = Read-Host 'What is the User account you looking for ?'
$dcs = Get-ADDomainController -Filter {Name -like "*"}
foreach($dc in $dcs)
{
    $hostname = $dc.HostName
    $user = Get-ADUser $userName -Server $hostname -Properties
lastLogon
    $lngexpires = $user.lastLogon
    if (-not ($lngexpires)) {$lngexpires = 0 }
    If ((($lngexpires -eq 0) -or ($lngexpires -gt
[DateTime]::.MaxValue.Ticks))
    {
        $LastLogon = "User Never Logged In"
    }
    Else
    {
        $Date = [DateTime]$lngexpires
        $LastLogon = $Date.AddYears(1600).ToLocalTime()
    }
}
Write-Host $username "last logged on at:" $LastLogon
```

The preceding script will ask for the username of the account and, once it is provided, the system will search for the lastLogon attribute value on all available domain controllers. If it cannot be found, it will return User Never Logged In or, if found, it will return the last logon timestamp.

Last login date report

Periodic housekeeping in AD is required for integrity. There may be user objects that have not been used for years. If we can create a report along with the last login dates, we can use it as a reference to clean up objects:

```
## Script For Filter user with Last logon Time ##
$htmlformat = "<style>BODY{background-color:LightBlue;}</style>"
Get-ADUser -Filter * -Properties "LastLogonDate" | sort-object -
property lastlogondate -descending | Select-Object Name,LastLogonDate
| ConvertTo-HTML -head $htmlformat -body "<H2>AD Accounts Last Login
```

```
Date</H2>" | Out-File C:\lastlogon.html
Invoke-Expression C:\lastlogon.html
```

This script creates an HTML report that includes all the user accounts with their last login date timestamps:

Name	LastLogonDate
admin2	24/04/2017 10:05:14
Adam	24/04/2017 08:01:29
Administrator	22/04/2017 19:53:30
AAD_d3cc81821dc8	15/04/2017 07:58:33
Scott Brewer	12/04/2017 01:49:31
liam	06/04/2017 23:46:22
Peter	06/04/2017 22:45:17
AD RMS Service	06/04/2017 19:48:18
Dishan Francis	06/04/2017 09:25:12
krbtgt	
Guest	
krbtgt_19295	
Inet User1	
_TechSupport_Template	
Dishan Francis	
Dale	
Test8 User8	
Test7 User7	
Test6 User6	
Test5 User5	
Test4 User4	
Test3 User3	
Test2 User2	
UserA	

Figure 17.4: Last login date HTML report

Some of the accounts in the above reports don't show the last login date value. It means no one has logged into those accounts yet.

Login failures report

It is important to know about failed attempts to log in to the DC, not just the successful attempts. These can be a result of potentially malicious activity.

The following script will create a report to indicate the login failures on a given domain controller:

```
## Report for DC login Failures ##
$failedevent = $null
$Date= Get-date
$dcc = Read-Host 'What is the Domain Controller ?'
$Report= "C:\auditreport.html"
$HTML=@"
<title>Failed Login Report for $dc</title>
<style>
BODY{background-color :LightBlue}
</style>
```

```

"@
$failedevent = Get-Eventlog security -Computer $dc -InstanceId 4625 -
After (Get-Date).AddDays(-7) |
Select TimeGenerated,ReplacementStrings |
% {
New-Object PSObject -Property @{
SourceComputer = $_.ReplacementStrings[13]
UserName = $_.ReplacementStrings[5]
SourceIPAddress = $_.ReplacementStrings[19]
Date = $_.TimeGenerated
}
}
$failedevent | ConvertTo-Html -Property
SourceComputer,UserName,SourceIPAddress,Date -head $HTML -body
"<H2>Failed Login Report for $dc</H2>" |
Out-File $Report
Invoke-Expression C:\auditreport.html

```

The aforementioned script is displayed in a way that's easy for readers to understand. When it is used in PowerShell, make sure to prevent extra line spaces.

When you run the preceding script, it will ask for the name of the domain controller that you wish to run this report against.

Then, in the background, it will search for event 4625 in the event viewer and then list the following data in a report:

- The source computer
- The username
- The source IP address
- The event time

The following screenshot shows the failed login report for REBEL-PDC-01:

SourceComputer	UserName	SourceIPAddress	Date
REBEL-PDC-01	administrator	127.0.0.1	24/04/2017 10:54:01
REBEL-PDC-01	administrator	127.0.0.1	24/04/2017 08:25:42
REBEL-PDC-01	administrator	127.0.0.1	24/04/2017 07:57:15
-	REBEL-SRV01\\$ -		23/04/2017 14:02:29
-	REBEL-SRV01\\$ -		23/04/2017 13:34:50
-			23/04/2017 13:33:47
-	REBEL-SRV01\\$ 192.168.0.131	192.168.0.131	23/04/2017 11:25:14
-		192.168.0.131	23/04/2017 09:55:37
REBEL-PDC-01	administrator	127.0.0.1	22/04/2017 20:24:35
REBEL-PDC-01	administrator	127.0.0.1	22/04/2017 19:53:26
REBEL-PDC-01	administrator	127.0.0.1	22/04/2017 18:05:24
REBEL-PDC-01	administrator	127.0.0.1	22/04/2017 13:17:39
REBEL-PDC-01	administrator	192.168.0.105	22/04/2017 11:10:22
REBEL-PDC-01	administrator	192.168.0.105	22/04/2017 11:08:22
REBEL-PDC-01	administrator	127.0.0.1	22/04/2017 11:07:39
REBEL-PDC-01	administrator	192.168.0.105	22/04/2017 11:06:21
REBEL-PDC-01	administrator	192.168.0.105	22/04/2017 11:04:21

Figure 17.5: Login failures report

The login failures records are different from one server to another. So when it comes to troubleshooting, make sure you select the correct domain controller.

Finding the locked-out account

If password policies are defined, accounts with a large number of login failures will be locked out. Locked-out accounts in an AD environment can be found using the following command:

```
Search-ADAccount -Lockedout | Select name,samAccountName,Lockedout
```

If any of those in the list need to be unlocked, we can use the `Unlock-ADAccount` cmdlet to unlock an account.

For an individual account, perform the following command:

```
Unlock-ADAccount tuser4
```

For all the accounts on the list, perform the following command:

```
Search-ADAccount -Lockedout | Unlock-ADAccount
```

It is not a good practice to unlock all the accounts unless there is a specific reason.

Password expire report

Issues due to expired passwords are a common support call type for helpdesks. The following script can generate a report about expiring passwords:

```

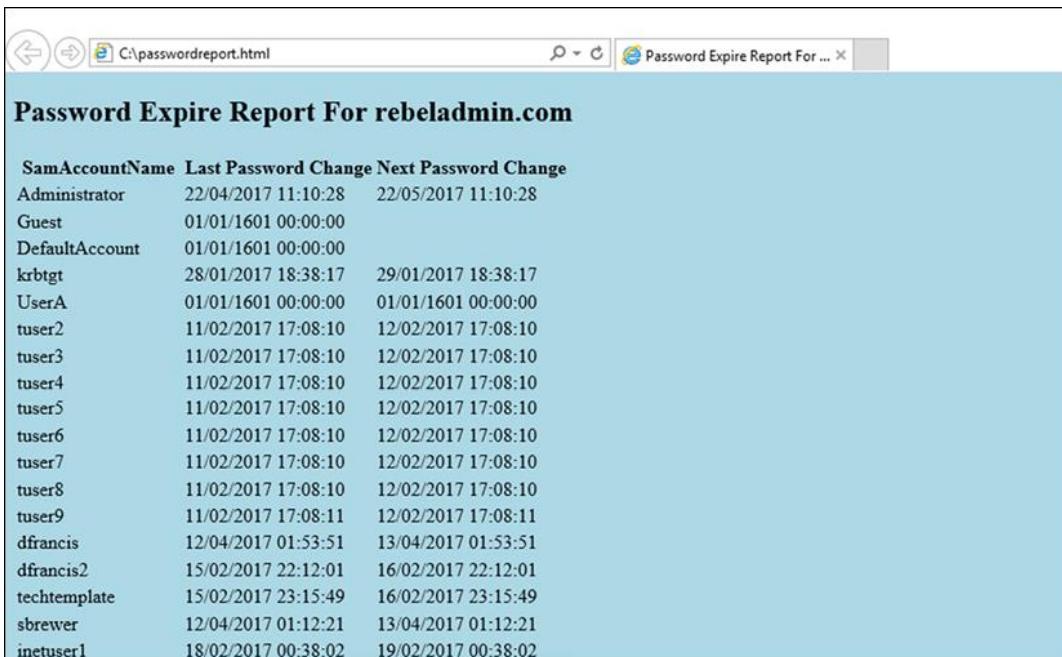
## Password Expire Report ##
$passwordreport = $null
$dc = (Get-ADDomain | Select DNSRoot).DNSRoot
$Report= "C:\passwordreport.html"
$HTML=@"
<title>Password Expire Report For $dc</title>
<style>
BODY{background-color :LightBlue}
</style>
"@
$passwordreport = Get-ADUser -filter * -Properties
"SamAccountName","pwdLastSet","msDS-UserPasswordExpiryTimeComputed" |
Select-Object -Property "SamAccountName",@{Name="Last Password
Change";Expression={ [datetime]::FromFileTime($_."pwdLastSet") }},@{Name
="Next Password Change";Expression={ [datetime]::FromFileTime($_."msDS-
UserPasswordExpiryTimeComputed") } }
$passwordreport | ConvertTo-Html -Property "SamAccountName","Last
Password Change","Next Password Change"-head $HTML -body "<H2>Password
Expire Report For $dc</H2>" |
Out-File $Report
Invoke-Expression C:\passwordreport.html

```

The aforementioned script is displayed in a way that's easy for readers to understand. When it is used in PowerShell, make sure to prevent extra line spaces.

This script will search for the attribute values for `SamAccountName`, `pwdLastSet`, and `msDS-UserPasswordExpiryTimeComputed` in every user object.

Then, they will be presented in an HTML report:



SamAccountName	Last Password Change	Next Password Change
Administrator	22/04/2017 11:10:28	22/05/2017 11:10:28
Guest	01/01/1601 00:00:00	
DefaultAccount	01/01/1601 00:00:00	
krbtgt	28/01/2017 18:38:17	29/01/2017 18:38:17
UserA	01/01/1601 00:00:00	01/01/1601 00:00:00
tuser2	11/02/2017 17:08:10	12/02/2017 17:08:10
tuser3	11/02/2017 17:08:10	12/02/2017 17:08:10
tuser4	11/02/2017 17:08:10	12/02/2017 17:08:10
tuser5	11/02/2017 17:08:10	12/02/2017 17:08:10
tuser6	11/02/2017 17:08:10	12/02/2017 17:08:10
tuser7	11/02/2017 17:08:10	12/02/2017 17:08:10
tuser8	11/02/2017 17:08:10	12/02/2017 17:08:10
tuser9	11/02/2017 17:08:11	12/02/2017 17:08:11
dfrancis	12/04/2017 01:53:51	13/04/2017 01:53:51
dfrancis2	15/02/2017 22:12:01	16/02/2017 22:12:01
techtemplate	15/02/2017 23:15:49	16/02/2017 23:15:49
sbrewer	12/04/2017 01:12:21	13/04/2017 01:12:21
inetuser1	18/02/2017 00:38:02	19/02/2017 00:38:02

Figure 17.6: Password expire report

All these reports can run as scheduled jobs and were developed to be sent over as an email every week or month. This saves administrators time and also prevents mistakes that can occur with manual tasks.

Review the membership of the high-level administrative groups

As a security best practice, it is important to limit the number of privileged accounts used in an Active Directory environment. Sometimes we add users to privileged groups temporarily to do certain tasks and then forget to remove the permissions later on. Therefore, it is important to review members of the sensitive groups periodically and update those as required. In Active Directory, the following security groups are identified as sensitive groups:

- Enterprise Admins
- Schema Admins
- Domain Admins
- Account Operators (if present)
- Server Operators (if present)
- Print Operators (if present)
- DHCP Administrators
- DNSAdmins

To review the membership of a sensitive group we can use the following command:

```
Get-ADGroupMember -Identity "Domain Admins"
```

In the preceding command, Domain Admins is the group name and it can be replaced with any other sensitive group name. Once we have the list of members, the next step is to find out if those accounts are actively used. To do that we can use the value of the LastLogonDate user attribute. By considering all of the above requirements, I created the following scripts to list down all the sensitive groups with their members and `LastLogonDate` values:

```
## Sensitive Group Report ##
$HTML=@"
<title>Sensitive Groups Membership Report</title>
<style>
BODY{background-color :LightBlue}
</style>
"@
$enterpriseadmins = Get-ADGroupMember -Identity "Enterprise Admins" |
where {$_.objectclass -eq 'user'} | Get-ADUser -Properties
LastLogonDate | select Name,LastLogonDate | ConvertTo-HTML -Property
```

```

"Name", "LastLogonDate" -Fragment -PreContent "<h2>Enterprise
Admins</h2>"  

$schemaadmins = Get-ADGroupMember -Identity "Schema Admins" | where
{$_ .objectclass -eq 'user'} | Get-ADUser -Properties LastLogonDate |
select Name,LastLogonDate | ConvertTo-HTML -Property  

"Name", "LastLogonDate" -Fragment -PreContent "<h2>Schema Admins</h2>"  

$domainadmins = Get-ADGroupMember -Identity "Domain Admins" | where
{$_ .objectclass -eq 'user'} | Get-ADUser -Properties LastLogonDate |
select Name,LastLogonDate | ConvertTo-HTML -Property  

"Name", "LastLogonDate" -Fragment -PreContent "<h2>Domain Admins</h2>"  

$accountoperators = Get-ADGroupMember -Identity "Account Operators" |  

where {$_ .objectclass -eq 'user'} | Get-ADUser -Properties  

LastLogonDate | select Name,LastLogonDate | ConvertTo-HTML -Property  

"Name", "LastLogonDate" -Fragment -PreContent "<h2>Account
Operators</h2>"  

$serveroperators = Get-ADGroupMember -Identity "Server Operators" |  

where {$_ .objectclass -eq 'user'} | Get-ADUser -Properties  

LastLogonDate | select  

Name,LastLogonDate | ConvertTo-HTML -Property "Name", "LastLogonDate" -  

Fragment -PreContent "<h2>Server Operators</h2>"  

$printoperators = Get-ADGroupMember -Identity "Print Operators" |  

where {$_ .objectclass -eq 'user'} | Get-ADUser -Properties  

LastLogonDate | select Name,LastLogonDate | ConvertTo-HTML -Property  

"Name", "LastLogonDate" -Fragment -PreContent "<h2>Print
Operators</h2>"  

$dnsadmins = Get-ADGroupMember -Identity "DnsAdmins" | where
{$_ .objectclass -eq 'user'} | Get-ADUser -Properties LastLogonDate |
select Name,LastLogonDate | ConvertTo-HTML -Property  

"Name", "LastLogonDate" -Fragment -PreContent "<h2>DNS Admins</h2>"  

$Reportvalues = ConvertTo-HTML -Body "$enterpiseadmins $schemaadmins  

$domainadmins $accountoperators $serveroperators $printoperators  

$dnsadmins" -Head $HTML  

$Reportvalues | Out-File "C:\sensativegroupreport.html"

```

The preceding script will produce an HTML report similar to below.

Enterprise Admins	
Name	LastLogonDate
Adalbert Edelmann	
dfrancis	7/15/2021 3:52:40 PM
Schema Admins	
Name	LastLogonDate
Anneluise Hildenbrand	
dfrancis	7/15/2021 3:52:40 PM
Domain Admins	
Name	LastLogonDate
dfrancis	7/15/2021 3:52:40 PM
SamiraA	7/16/2021 10:13:30 AM
Account Operators	
Name	LastLogonDate
Ulla Mueller	
Server Operators	
Name	LastLogonDate
Irmtrud Kutzner	
Print Operators	
Name	LastLogonDate
Adalbert Kettler	
DNS Admins	
Name	LastLogonDate
Judith Markus	

Figure 17.7: Sensitive group memberships

In this report, the LastLogonDate value can be used as a reference and if the account is not used often, we can go ahead and remove it from the sensitive groups.

In the script, I didn't mention DHCP Administrators as it is a local group.

In the above report, we are listing all the users in sensitive groups. Then it is up to engineers to tidy up the group membership. However, we can further develop this script and list down users who haven't logged in for a certain number of days. Then engineers can use it as a starting point to update the group memberships.

```
## Sensitive Group Members Inactive for 30 days ##
$30Days = (get-date).adddays(-30)
$HTML=@"
<title>Sensitive Groups Membership Report : USers Inactive for 30
days</title>
<style>
BODY{background-color :LightBlue}
</style>
"@
$enterpriseadmins = Get-ADGroupMember -Identity "Enterprise Admins" |
where {$_.objectclass -eq 'user'} | Get-ADUser -Properties
```

```

LastLogonDate |Where {$_.LastLogonDate -le $30Days} | select
Name,LastLogonDate | ConvertTo-HTML -Property "Name","LastLogonDate" -
Fragment -PreContent "<h2>Enterprise Admins</h2>"
$schemaadmins = Get-ADGroupMember -Identity "Schema Admins" | where
{$_.objectclass -eq 'user'} | Get-ADUser -Properties LastLogonDate |
Where {$_.LastLogonDate -le $30Days}| select Name,LastLogonDate |
ConvertTo-HTML -Property "Name","LastLogonDate" -Fragment -PreContent
"<h2>Schema Admins</h2>"
$domainadmins = Get-ADGroupMember -Identity "Domain Admins" | where
{$_.objectclass -eq 'user'} | Get-ADUser -Properties LastLogonDate |
Where {$_.LastLogonDate -le $30Days}| select Name,LastLogonDate |
ConvertTo-HTML -Property "Name","LastLogonDate" -Fragment -PreContent
"<h2>Domain Admins</h2>"
$accountoperators = Get-ADGroupMember -Identity "Account Operators" |
where {$_.objectclass -eq 'user'} | Get-ADUser -Properties
LastLogonDate | Where {$_.LastLogonDate -le $30Days}| select
Name,LastLogonDate | ConvertTo-HTML -Property "Name","LastLogonDate" -
Fragment -PreContent "<h2>Account Operators</h2>"
$serveroperators = Get-ADGroupMember -Identity "Server Operators" |
where {$_.objectclass -eq 'user'} | Get-ADUser -Properties
LastLogonDate | Where {$_.LastLogonDate -le $30Days}| select
Name,LastLogonDate | ConvertTo-HTML -Property "Name","LastLogonDate" -
Fragment -PreContent "<h2>Server Operators</h2>"
$printoperators = Get-ADGroupMember -Identity "Print Operators" |
where {$_.objectclass -eq 'user'} | Get-ADUser -Properties
LastLogonDate | Where {$_.LastLogonDate -le $30Days}| select
Name,LastLogonDate | ConvertTo-HTML -Property "Name","LastLogonDate" -
Fragment -PreContent "<h2>Print Operators</h2>"
$dnsadmins = Get-ADGroupMember -Identity "DnsAdmins" | where
{$_.objectclass -eq 'user'} | Get-ADUser -Properties LastLogonDate |
Where {$_.LastLogonDate -le $30Days}| select Name,LastLogonDate |
ConvertTo-HTML -Property "Name","LastLogonDate" -Fragment -PreContent
"<h2>DNS Admins</h2>"
$Reportvalues = ConvertTo-HTML -Body "$enterpiseadmins $schemaadmins
$domainadmins $accountoperators $serveroperators $printoperators
$dnsadmins" -Head $HTML
$Reportvalues | Out-File "C:\inactiveusers.html"

```

In the above, I am creating an HTML report that lists down the members of sensitive groups who haven't logged in for the last 30 days.

Dormant accounts

In Active Directory, at least 10% of user accounts are dormant (inactive) accounts. These accounts can represent:

- Test accounts
- Contractors

- Former employees
- Disabled accounts

It is important to review these dormant accounts periodically and remove all unnecessary accounts from Active Directory as they are a possible security threat. If it is not possible to remove some of these accounts, at least remove them from sensitive groups and disable the accounts.

We can find these accounts in Active Directory by looking at the `LastLogonDate` attribute value and the account status. By considering these requirements, I created the following script to find dormant accounts:

```
## Dormant Accounts ##
$InactiveDate = (Get-Date) .AddDays (-30)
$HTML=@"
<title>Dormant Accounts Report</title>
<style>
BODY{background-color :LightBlue}
</style>
"@
$disabledaccounts = Get-ADUser -Filter {Enabled -eq $false} | select
samAccountName,GivenName,Surname | ConvertTo-Html -Property
"samAccountName","GivenName","Surname" -Fragment -PreContent
"<h2>Disabled Account</h2>"
$inactiveaccounts = Get-ADUser -Filter {LastLogonDate -lt
$InactiveDate -and Enabled -eq $true} -Properties LastLogonDate |
select samAccountName,GivenName,Surname,LastLogonDate | ConvertTo-Html
-Property "samAccountName","GivenName","Surname","LastLogonDate" -
Fragment -PreContent "<h2>Inactive Accounts</h2>"
$Reportvalues = ConvertTo-HTML -Body "$disabledaccounts
$inactiveaccounts" -Head $HTML
$Reportvalues | Out-File "C:\dormantusers.html"
```

In the above, I am looking for disabled accounts by using `Get-ADUser -Filter {Enabled -eq $false} | select samAccountName,GivenName,Surname`. Then, after that, I am searching for users who have been inactive for the last 30 days by using `Get-ADUser -Filter {LastLogonDate -lt $InactiveDate -and Enabled -eq $true} -Properties LastLogonDate | select samAccountName,GivenName,Surname,LastLogonDate`. Here, I am ignoring disabled accounts as they are already recorded as separate entities. At the end, I am passing the findings to an HTML report called `dormantusers.html`.

Disabled Account

SamAccountName	GivenName	Surname
krbtgt		
RA756230	Edelhard	Brenner
RA540628	Edelbert	Hörl
RA274595	Edda	Ruck
RA212072	Edelfried	Riemer
RA939522	Eddi	Munz
RA668519	Edelinde	Bönisch
RA761754	Edelinde	Stich
Guest		

Inactive Accounts

SamAccountName	GivenName	Surname	LastLogonDate
dfrancis			7/15/2021 3:52:40 PM
MSOL_3982d7047229			7/15/2021 4:32:21 PM
RonHD			7/16/2021 9:12:19 AM
JeffL			7/16/2021 9:37:09 AM
SamiraA			7/16/2021 10:13:30 AM

Figure 17.8: Dormant accounts

If required, this data can also be exported into a CSV file. But in these examples, I used the HTML format to show the results in a user-friendly way.

Users with the Password Never Expires setting

In an Active Directory environment, we use password policies to enforce users to follow complexity standards and other best practices related to passwords. Users should use complex passwords and should update their passwords at regular intervals. This is one of the basic requirements of identity protection. However, if the user account has the Password Never Expires setting enabled, the user will not be forced to update the passwords according to the password policy.

We can find Active Directory user accounts that have the Password Never Expires setting enabled by using the following PowerShell commands:

```
Get-ADUser -Filter {passwordNeverExpires -eq $true -and Enabled -eq $true } -Properties * | Select samAccountName,GivenName,Surname
```

In the preceding command, I am looking for the `passwordNeverExpires` attribute value and if it's set to true, it means the setting is enabled. At the same time, I also checked if the user account is active.



```
Administrator: PowerShell 7 (x64)
PowerShell 7.1.4
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.

PS C:\Users\dfrancis> Get-ADUser -Filter {passwordNeverExpires -eq $true -and Enabled -eq $true } -Properties * | Select samAccountName,GivenName,Surname

samAccountName GivenName Surname
----- -----
tuser2      Test User 2
dfrancis    Dishan     Francis
adam        adam       adam

PS C:\Users\dfrancis>
```

Figure 17.9: Users with the Password Never Expires setting

In the next section of this chapter, we are going to look into managing Active Directory objects in a hybrid environment.

Azure Active Directory PowerShell

Similar to on-prem Active Directory, we also can use PowerShell to manage Azure Active Directory.

Let's see why we should use PowerShell to manage Azure Active Directory:

- **Early bird access to features:** Microsoft keeps releasing new features, bug fixes, updates, and feature enhancements more frequently to Azure AD services than on-prem Active Directory.
Microsoft releases new features to the public in two stages. In the first stage, they are released as a preview version. This is not recommended for use in production, but IT professionals can use them for testing and provide feedback to Microsoft. At this stage, the features can have many updates and, most of the time, it will take some time to update the GUI accordingly. Some of these changes will not be available on the GUI until general release. But if we are using PowerShell, we do not have to wait. We can have early access to features as soon as they are released.
- **Faster response:** The Azure Active Directory portal has many different windows, wizards, and forms to configure and manage users, groups, roles, and associated features. The GUI makes it easy to do things, but it takes time. As an example, if you add a user account using the Azure AD portal, you have to go to four sub-windows at least. But PowerShell allows us to do it using one window and a few lines of commands.
- **Granular control:** The Azure AD portal visualizes the data and configuration of the service using different windows. However, it may not always show what we want. As an example, let's assume we are looking for a specific value in two user accounts. If we use the GUI, we need to go to a few different windows to gather this information. But using a PowerShell command or script, we will be able to gather the same information in one window. This is really helpful when troubleshooting.
- **Microsoft Graph integration:** Microsoft Graph provides a unified programmability model to access a vast amount of data in Microsoft 365, Azure Active Directory, Enterprise Mobility Suite,

Windows 10, and so on. As part of it, the Azure AD PowerShell for Graph module allows you to retrieve data, update directory configurations, add/update/remove objects, and configure features via Microsoft Graph.

In this chapter, I will be using the Azure Active Directory PowerShell for Graph module to manage an Azure AD hybrid environment.

Installation

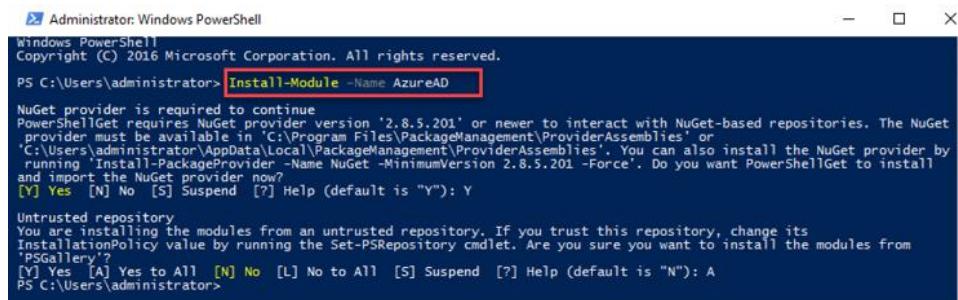
The Azure Active Directory PowerShell for Graph module comes in two versions. The public preview version is the most recent, but it is not recommended for use in production.

The installation steps for this version can be found at <https://bit.ly/3HR3EpU>.

The general availability version is the stable, recommended version for production environments. It can be installed on any computer that runs Windows Server 2008 R2 or above with the latest updates. Microsoft .NET Framework 4.5 or above is also required.

Once the prerequisites are in place, perform the following steps:

1. Log in to the computer you have selected for the Azure Active Directory PowerShell for Graph module.
2. Launch the PowerShell console as an administrator.
3. Run the `Install-Module -Name AzureAD` command. Answer Yes if it is a required repository update:



```
Administrator: Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\administrator> Install-Module -Name AzureAD

NuGet provider is required to continue
PowerShellGet requires NuGet provider version '2.8.5.201' or newer to interact with NuGet-based repositories. The NuGet provider must be available in 'C:\Program Files\PackageManagement\ProviderAssemblies' or
'C:\Users\administrator\AppData\Local\PackageManagement\ProviderAssemblies'. You can also install the NuGet provider by
running 'Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force'. Do you want PowerShellGet to install
and import the NuGet provider now?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): Y

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from
'PSGallery'?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): A
PS C:\Users\administrator>
```

Figure 17.10: Install AzureAD PowerShell module

4. After installation, we can verify the module installation using `Get-Module AzureAD`.
5. After successfully installing the module, run `Connect-AzureAD` to initiate a connection to the Azure AD tenant.
6. Then, it will prompt you with a login window. Use Azure AD global administrator account details to connect.

Now we have the Azure Active Directory PowerShell for Graph module installed. Let's see how we can manage an Azure AD hybrid environment using this module.

Azure AD and MSOL modules are not supported in PowerShell 7.x. Therefore, here, I am using default Windows PowerShell running on Windows 10.

General commands

We can start by listing all the available commands under the Azure AD module, which can be done by using the following:

```
Get-Command -module AzureAD
```

We can view the full syntax for a command by using the `Get-Help` command. As an example, we can view the full syntax for the `Get-AzureADUser` command using the following:

```
Get-Help Get-AzureADUser
```

We can verify the status of Azure AD domains using the following command:

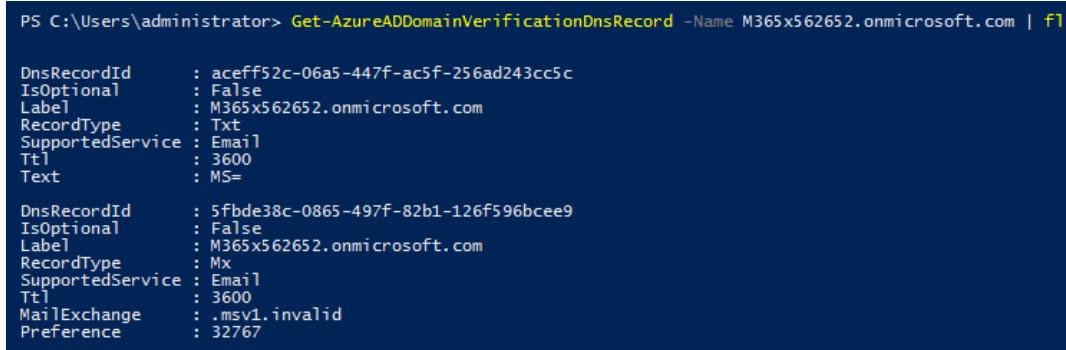
```
Get-AzureADDomain | fl
```

The preceding command helps to identify the domain verification status by referring to the value of the `IsVerified` attribute.

If you are using a custom domain in Azure AD, we need to verify ownership of the domain using DNS records. If it is not verified, we can retrieve the required DNS records by using the following command:

```
Get-AzureADDomainVerificationDnsRecord -Name M365x562652.onmicrosoft.com | fl
```

In the preceding example, `M365x562652.onmicrosoft.com` represents the domain name:



PS C:\Users\administrator> Get-AzureADDomainVerificationDnsRecord -Name M365x562652.onmicrosoft.com | fl

DnsRecordId	IsOptional	Label	RecordType	SupportedService	Ttl	Text
aceff52c-06a5-447f-ac5f-256ad243cc5c	: False	: M365x562652.onmicrosoft.com	: Txt	: Email	: 3600	: MS=
5fbde38c-0865-497f-82b1-126f596bcee9	: False	: M365x562652.onmicrosoft.com	: Mx	: Email	: 3600	: .msv1.invalid
				MailExchangePreference	: 32767	

Figure 17.11: Required DNS records for domain verifications

We can view the details of the Azure AD tenant by using the following:

```
Get-AzureADTenantDetail | fl
```

In a hybrid environment, the health of the on-prem AD sync is crucial. We can view the time of the last directory sync by using the following command:

```
Get-AzureADTenantDetail | select CompanyLastDirSyncTime
```

Managing users

We can view the user account details for a known account using the following:

```
Get-AzureADUser -ObjectId AdeleV@M365x562652.OnMicrosoft.com | fl
```

In the preceding command, `AdeleV@M365x562652.OnMicrosoft.com` represents the UPN of the user.

We also can use user attributes to find user account details:

```
Get-AzureADUser -Filter "startswith(GivenName, 'Adele')"
```

The preceding command will filter Azure AD users with `GivenName` as Adele.

We can also filter users based on a specific attribute value:

```
Get-AzureADUser -Filter "GivenName eq 'Adele'"
```

The preceding command will search for the exact user with the given name value Adele.

In my demo environment, I'd like to see a list of disabled accounts. I can do this using the following command:

```
Get-AzureADUser -All $true -Filter 'accountEnabled eq false'
```

We can modify the output of the filtered data further:

```
Get-AzureADUser -All $true -Filter 'accountEnabled eq false' | select DisplayName, UserPrincipalName, Department
```

The preceding command will display the value of the `DisplayName`, `UserPrincipalName`, and `Department` attributes of the filtered accounts.

In a hybrid environment, we can filter accounts that are synced from on-prem AD by using the following:

```
Get-AzureADUser -All $true -Filter 'DirSyncEnabled eq true'
```

In the preceding command, the value of the `DirSyncEnabled` attribute defines whether it's a cloud-only account or a synced account.

We also can check the last sync value for the synced accounts:

```
Get-AzureADUser -All $true -Filter 'DirSyncEnabled eq true' | select DisplayName, UserPrincipalName, LastDirSyncTime
```

In the preceding command, the `LastDirSyncTime` value defines the last sync time of the object.

We can also export the output to a CSV file using the `Export-CSV` command:

```
Get-AzureADUser -All $true -Filter 'DirSyncEnabled eq true' | select DisplayName, UserPrincipalName, LastDirSyncTime | Export-CSV -Path .\syncaccount.csv
```

The ImmutableId value of a user account is used to map an Azure AD user object to an on-prem user object. ImmutableId does have a relationship with on-prem user accounts' ObjectGUID. We can use this to identify cloud-only users. If it is a cloud-only user, the ImmutableId value should be null:

```
Get-AzureADUser -All $true | where-Object {$_._.ImmutableId -eq $null}
```

The preceding command returns a list of all the cloud-only accounts. We can export the required attribute values to CSV by using the following:

```
Get-AzureADUser -All $true | where-Object {$_._.ImmutableId -eq $null} | select DisplayName,UserPrincipalName | Export-Csv -Path .\cloudaccount.csv
```

Another important thing related to accounts is licences. If we are going to use Azure AD's premium features, we need to have relevant licenses assigned. By default, a user only has Azure AD free version features.

To view licenses associated with a user account, we can use the following command:

```
Get-AzureADUserLicenseDetail -ObjectId MeganB@M365x562652.OnMicrosoft.com | fl
```

The preceding command will return the licenses associated with the user **MeganB@M365x562652.OnMicrosoft.com**.

We also can view the subscribed SKUs using the following command:

```
Get-AzureADSubscribedSku | fl
```

The preceding command lists all the details about licenses that are associated with the tenant. However, we only need to know how many licenses have been used and how many licenses are available. We can do this using the following command:

```
Get-AzureADSubscribedSku | select SkuPartNumber,ConsumedUnits -ExpandProperty PrepaidUnits
```

In the preceding example, the **SkuPartNumber** value represents the license part number. The value of the enabled field represents the number of purchased licenses. **ConsumedUnits** represents the number of consumed licenses.

Let's move on and see how we can assign a new license to a user.

In my environment, I have a user who synced from on-prem Azure AD who doesn't have a license assigned:

```
Get-AzureADUserLicenseDetail -ObjectId ADJellison@M365x562652.onmicrosoft.com | fl
```

The following screenshot displays the output of the preceding command:



C:\Users\administrator> Get-AzureADUserLicenseDetail -ObjectId ADJellison@M365x562652.onmicrosoft.com | fl
C:\Users\administrator>

Figure 17.12: Check Azure AD license assignment

As a first step, let's create objects to use in the license assignment process:

```
$newlicence = New-Object -TypeName  
Microsoft.Open.AzureAD.Model.AssignedLicense  
$newlicenceadd = New-Object -TypeName  
Microsoft.Open.AzureAD.Model.AssignedLicenses
```

Then, we need to find the **SkuId** of the licenses.

I am going to assign the **ENTERPRISEPREMIUM** license to the user:

```
$newlicence.SkuId = (Get-AzureADSubscribedSku | Where-Object -Property  
SkuPartNumber -Value "ENTERPRISEPREMIUM" -EQ) .Skuid
```

Then, we need to assign the licenses to the object:

```
$newlicenceadd.AddLicenses = $newlicence
```

Now, we can go ahead and assign the license to the user:

```
Set-AzureADUserLicense -ObjectId  
"ADJellison@M365x562652.onmicrosoft.com" -AssignedLicenses  
$newlicenceadd
```

The preceding command assigns **ENTERPRISEPREMIUM** licenses to the user

ADJellison@M365x562652.onmicrosoft.com:

```
PS C:\Users\administrator> Get-AzureADUserLicenseDetail -ObjectId ADJellison@M365x562652.onmicrosoft.com | fl  
  
ObjectId      : irYR8iuwxk5Axcc9WXUe7mAn38eBLPd0tXhbU5K1cd8  
ServicePlans  : {  
    class ServicePlanInfo {  
        AppliesTo: Company  
        ProvisioningStatus: PendingProvisioning  
        ServicePlanId: 4a51bcfa5-1eff-43f5-878c-177680f191af  
        ServicePlanName: WHITEBOARD_PLAN  
    }  
    class ServicePlanInfo {  
        AppliesTo: Company  
        ProvisioningStatus: Success  
        ServicePlanId: efb0351d-3b08-4503-993d-383af8de41e3  
        ServicePlanName: MIP_S_CLP2  
    }  
    class ServicePlanInfo {  
        AppliesTo: Company  
        ProvisioningStatus: Success  
        ServicePlanId: 5136a095-5cf0-4aff-bec3-e84448b38ea5  
        ServicePlanName: MIP_S_CLP1  
    }  
    class ServicePlanInfo {  
        AppliesTo: Company  
        ProvisioningStatus: Success  
        ServicePlanId: 33c4f319-9bdd-48d6-9c4d-410b750a4a5a  
        ServicePlanName: MYANALYTICS_P2  
    }  
...}  
SkuId         : c7df2760-2c81-4ef7-b578-5b5392b571df  
SkuPartNumber : ENTERPRISEPREMIUM
```

Figure 17.13: Assign license to user

It is a must to set the **UsageLocation** value for users who sync from on-prem AD before assigning licenses. We can do this using **Set-AzureADUser -ObjectId ADJellison@M365x562652.onmicrosoft.com -UsageLocation "US"**.

We can remove the licenses assigned using the following command:

```
$licenseB = New-Object -TypeName
Microsoft.Open.AzureAD.Model.AssignedLicenses
$licenseB.RemoveLicenses = (Get-AzureADSubscribedSku | Where-Object
{$_ .SkuPartNumber -eq 'ENTERPRISEPREMIUM'}).Skuid
Set-AzureADUserLicense -ObjectId
"ADJellison@M365x562652.onmicrosoft.com" -AssignedLicenses $licenseB
```

Using the preceding commands, I have created a script to do the following:

- Search for users who synced from on-prem AD
- Of those users, select the users who don't have Azure AD licenses assigned
- Set the UsageLocation value for selected users
- Assign Azure AD licenses to selected users

```
#####Script to Assign Licences to Synced Users from On-Permisises
AD#####
Import-Module AzureAD
Connect-AzureAD
####Filter Synced Users who doesn't have licence assigned#####
$ADusers = Get-AzureADUser -All $true -Filter 'DirSyncEnabled eq true'
$notlicenced = Get-AzureADUser -All $true | Where-Object
{$ADusers.AssignedLicenses -ne $null} | select ObjectId | Out-File -
FilePath C:\users.txt
#####Set UsageLocation value to sync users#####
(Get-Content "C:\users.txt" | select-object -skip 3) | ForEach { Set-
AzureADUser -ObjectId $_ -UsageLocation "US" }
#####Set User Licecnes#####
$newlicence = New-Object -TypeName
Microsoft.Open.AzureAD.Model.AssignedLicense
$newlicenceadd = New-Object -TypeName
Microsoft.Open.AzureAD.Model.AssignedLicenses
$newlicence.Skuid = (Get-
AzureADSubscribedSku | Where-Object -Property SkuPartNumber -Value
"ENTERPRISEPREMIUM" -EQ).Skuid
$newlicenceadd.AddLicenses = $newlicence
(Get-Content "C:\users.txt" | select-object -skip 3) | ForEach { Set-
AzureADUserLicense -ObjectId $_ -AssignedLicenses $newlicenceadd }
```

In a hybrid environment, users are mainly created through on-prem Active Directory, but there are occasions when we need to add cloud-only accounts. This is mainly for cloud management tasks.

We can create a new user by using the following command:

```
$Userpassword = New-Object -TypeName
Microsoft.Open.AzureAD.Model.PasswordProfile
$Userpassword.Password = "London@1234"
New-AzureADUser -DisplayName "Andrew Xavier" -PasswordProfile
$Userpassword -UserPrincipalName
```

```
"Andrew.Xavier@M365x562652.onmicrosoft.com" -AccountEnabled $true -  
MailNickName "AndrewXavier"
```

In the preceding command, `-PasswordProfile` is used to define the password profile for the new user account. `-MailNickName` defines the value for the user's mail nickname. In the preceding example, add a new user account, `Andrew.Xavier@M365x562652.onmicrosoft.com`, with the password `London@1234`.

We also can create multiple user accounts using CSV files. In the following example, I am using a CSV file to create users. The CSV file contains the following:

```
UserPrincipalName, DisplayName, MailNickName  
DishanM@M365x562652.onmicrosoft.com, Dishan Melroy, DishanMel  
JackM@M365x562652.onmicrosoft.com, Jack May, JackMay  
RicahrdP@M365x562652.onmicrosoft.com, Richard Parker, RichardPar
```

Then, I can create these new users using the following:

```
$Userpassword = New-Object -TypeName  
Microsoft.Open.AzureAD.Model.PasswordProfile  
$Userpassword.Password = "London@1234"  
Import-Csv -Path C:\newuser.csv | foreach {New-AzureADUser -  
UserPrincipalName $_.UserPrincipalName -DisplayName $_.DisplayName -  
MailNickName $_.MailNickName -PasswordProfile $Userpassword -  
AccountEnabled $true}
```

By using the preceding commands, I have created a script to do the following:

- Create new user accounts using a CSV file
- Set `UsageLocation` for new user accounts
- Assign `ENTERPRISEPREMIUM` licenses to users

```
#####A Script to create new users and assign Azure AD  
licences#####  
Import-Module AzureAD  
Connect-AzureAD  
#####Create New Users using CSV #####  
$Userpassword = New-Object -TypeName  
Microsoft.Open.AzureAD.Model.PasswordProfile  
$Userpassword.Password = "London@1234"  
Import-Csv -Path C:\newuser.csv | foreach {New-AzureADUser -  
UserPrincipalName $_.UserPrincipalName -DisplayName $_.DisplayName -  
MailNickName $_.MailNickName -PasswordProfile $Userpassword -  
UsageLocation "US" -AccountEnabled $true} | select ObjectId | Out-File  
-FilePath C:\users.txt  
#####Assign Licences#####  
$newlicence = New-Object -TypeName  
Microsoft.Open.AzureAD.Model.AssignedLicense  
$newlicenceadd = New-Object -TypeName
```

```

Microsoft.Open.AzureAD.Model.AssignedLicenses
$newlicence.SkuId = (Get-AzureADSubscribedSku | Where-Object -Property
SkuPartNumber -Value "ENTERPRISEPREMIUM" -EQ).SkuId
$newlicenceadd.AddLicenses = $newlicence
(Get-Content "C:\users.txt" | select-object -skip 3) | ForEach { Set-
AzureADUserLicense -ObjectId $_ -AssignedLicenses $newlicenceadd }

```

To remove an Azure AD user, we can use the following:

```
Remove-AzureADUser -ObjectId "JDAllen@M365x562652.onmicrosoft.com"
```

We can combine it with a user search using the following command:

```
Get-AzureADUser -Filter "startswith(DisplayName,'Dishan')" | Remove-
AzureADUser
```

The preceding command will search for user accounts that have a `DisplayName` that starts with Dishan. If there are any, the second part of the command will remove them.

Managing groups

Azure AD groups also work similarly to on-prem AD groups. They can be used to manage permissions in an effective manner. In a hybrid environment, there will be cloud-only groups as well as synced groups from the on-prem AD environment. In this section, we are going to look into group management using the Azure Active Directory PowerShell for Graph module.

Let's start with listing groups. We can search for a group using the following command:

```
Get-AzureADGroup -SearchString "sg"
```

In the preceding command, `SearchString` is used to define the search criteria. The preceding example will list any groups containing sg in the `DisplayName` field:

PS C:\Users\administrator> Get-AzureADGroup -SearchString "sg"		
ObjectId	DisplayName	Description
93291438-be19-472e-a1d6-9b178b7ac619	sg-Engineering	All engineering personnel
2a11d3ee-8383-44d1-9fb9-85cb4dcc2d5a	sg-Executive	All executives
38bd48e7-e37c-48c9-bae4-7f00949529a0	sg-Finance	All finance personnel
c93737dd-0e77-4325-8d5e-6524d8baedff	sg-HR	All HR personnel
c00fc3df-a395-48fb-a620-05d64384fa3e	sg-IT	All IT personnel
f7ca523a-3a17-4755-aea9-5c723e8d9c6a	sg-Legal	All legal executives
73115583-7ce1-4890-8b96-40bc257e0186	sg-Operations	All operations personnel
0636227c-0a03-4584-a72d-52e5479c2aad	sg-Retail	All retail Users
6a232560-7a59-413c-8f2b-60ecdab21cb	sg-Sales and Marketing	All marketing personnel

Figure 17.14: Search for groups

In the search result, we can see the `ObjectId` for the group. Once we know the `ObjectId`, we can see the details of the group using the following command:

```
Get-AzureADGroup -ObjectId 93291438-be19-472e-a1d6-9b178b7ac619 | fl
```

In a hybrid environment, there will be security groups that have synced from the on-premises Active Directory. We can filter these groups using the following:

```
Get-AzureADGroup -Filter 'DirSyncEnabled eq true' | select  
ObjectId, DisplayName, LastDirSyncTime
```

In the preceding example, the `LastDirSyncTime` column displays the last successful sync time of the group.

We can filter cloud-only groups using the following command:

```
Get-AzureADGroup -All $true | where-Object  
{$_ .OnPremisesSecurityIdentifier -eq $null}
```

In the preceding command, we are using the `OnPremisesSecurityIdentifier` attribute to filter the groups. This attribute only has value if it is synced from on-premises AD.

We can view group memberships by using the following:

```
Get-AzureADGroupMember -ObjectId 2a11d5ee-8383-44d1-9fbd-85cb4dcc2d5a
```

In the preceding command, we are using `ObjectId` to uniquely identify the group.

We can add members to the group using the `Add-AzureADGroupMember` cmdlet:

```
Add-AzureADGroupMember -ObjectId 2a11d5ee-8383-44d1-9fbd-85cb4dcc2d5a  
-RefObjectId a6aec9-909e-4684-8712-d0f242451338
```

In the preceding command, the `ObjectId` value represents the group, and the `RefObjectId` value represents the user.

We can remove a member from the group by using the following command:

```
Remove-AzureADGroupMember -ObjectId 2a11d5ee-8383-44d1-9fbd-  
85cb4dcc2d5a -MemberId a6aec9-909e-4684-8712-d0f242451338
```

In the preceding command, the `ObjectId` value represents the group, and the `MemberId` value represents the user's `ObjectId`.

We can also combine the `Add-AzureADGroupMember` cmdlet with the `Get-AzureADUser` cmdlet to add bulk users to a group.

In the following script, I used the `Get-AzureADUser` cmdlet to search for users in Marketing Department, and then used `Add-AzureADGroupMember` to add those users to Sales Group as members:

```
#####Script to Add Multiple users to Security Group#####
Import-Module AzureAD
Connect-AzureAD
##### Search for users in Marketing Department #####
Get-AzureADUser -All $true -Filter "Department eq 'Marketing'" |
select ObjectId | Out-File -FilePath C:\salesusers.txt
#####Add Users to Sales Group#####

```

```
(Get-Content "C:\salesusers.txt" | select-object -skip 3) | ForEach {  
Add-AzureADGroupMember -ObjectId f9f51d29-e093-4e57-ad79-2fc5ae3517db  
-RefObjectId $_ }
```

In a hybrid environment, security groups are mainly synced from on-prem AD. But there can be requirements for cloud-only groups as well. We can create a cloud-only group by using the following:

```
New-AzureADGroup -DisplayName "REBELADMIN Sales Team" -MailEnabled  
$false -MailNickname "salesteam" -SecurityEnabled $true
```

The following screenshot displays the output of the preceding command:

ObjectId	DisplayName	Description
7592b555-343d-4f73-a6f1-2270d7cf014f	REBELADMIN Sales Team	

Figure 17.15: Create a cloud-only group

The preceding command creates a security group called REBELADMIN Sales Team. This group is not a mail-enabled group.

We can remove an Azure AD group using the following command:

```
Remove-AzureADGroup -ObjectId 7592b555-343d-4f73-a6f1-2270d7cf014f
```

In the preceding command, the ObjectId value defines the group.

Apart from security groups, Azure AD also has predefined administrative roles, which can be used to assign access permissions to Azure AD and other cloud services. There are more than 35 predefined administrative roles. Each role has its own set of permissions. More details about these roles can be found at <https://bit.ly/3r5FMcs>.

We can list all the administrative roles using the following:

```
Get-AzureADDirectoryRoleTemplate
```

By default, only a few administrative roles are enabled. We can list these roles using the following:

```
Get-AzureADDirectoryRole
```

Here, the company administrator directory role represents the Azure AD global administrators.

We can enable the administrative role using the following:

```
Enable-AzureADDirectoryRole -RoleTemplateId e6d1a23a-d411-4be4-9570-  
befc86d067a7
```

In the preceding command, the **RoleTemplateId** value represents the administrative role.

We can assign the administrative role to a user by using the following command:

```
Add-AzureADDirectoryRoleMember -ObjectId b63c1671-625a-4a80-8bae-  
6487423909ca -RefObjectId 581c7265-c8cc-493b-9686-771b2f10a77e
```

In the preceding command, the `ObjectId` value represents the administrative role. `RefObjectId` is the object ID value of the user.

We can list members of the administrative role using the following:

```
Get-AzureADDirectoryRoleMember -ObjectId 36b9ac02-9dfc-402a-8d44-  
ba2d8995dc06
```

In the preceding command, `ObjectId` represents the administrative role.

We can remove a member from the role using the following command:

```
Remove-AzureADDirectoryRoleMember -ObjectId 36b9ac02-9dfc-402a-8d44-  
ba2d8995dc06 -MemberId 165ebcb7-f07d-42d2-a52e-90f44e71e4a1
```

In the preceding command, `MemberId` is equal to the user's object ID value.

This marks the end of this section. There are lots of cmdlets that can still be used to manage Azure AD, but here I explained the cmdlets that will be required for day-to-day operations.

Microsoft Graph

Microsoft Graph is like a gateway that allows users to access enormous amounts of data and collect information from:

1. Microsoft 365 core services (for example, Office 365, Microsoft Search, OneDrive, SharePoint)
2. Identity and Security Services (for example, Azure AD, Defender 365, Endpoint Manager)
3. Windows 10 services

Microsoft Graph connects to the above services by using REST APIs and client libraries to retrieve required data.

We can use three methods to interact with Microsoft Graph data:

1. The Microsoft Graph API endpoint (<https://bit.ly/3DSbZHF>) can be used to access data and information collected from various Microsoft services. This data can be processed and present in the way an organization/individual requires. Also, this data can be used to develop rich applications/services.
2. Microsoft Graph connectors help to bring third-party application/service data to Microsoft Search so all company data can be searched from one location. Some of these connectors are built by Microsoft and others are built by partners. You can access the connectors gallery by using <https://bit.ly/3FMBaft>.
3. Microsoft Graph Data Connect allows us to access Microsoft Graph data at scale. This gives engineers a granular level of control over Microsoft Graph data. Microsoft Graph Data Connect also provides a unique set of tools that can be used to build intelligent applications.

Based on the requirements, we can use one or more of the above methods to access Microsoft Graph data.

In this section, I am going to demonstrate how we can use Microsoft Graph to access Azure AD data.

Microsoft Graph Explorer

Graph Explorer is a Microsoft-developed tool that allows you to make Microsoft Graph REST API requests. This tool can be accessed using <https://bit.ly/3I6EF8m>.

The first thing we need to do on the page is to log in. Then we need to grant permissions to Microsoft Graph to access data. To do that, go to the **Modify permissions (Preview)** tab and give consent to grant relevant permissions.

The screenshot shows the Microsoft Graph Explorer interface. In the top navigation bar, 'Solutions' is selected. Below it, 'Graph Explorer' is the active tab. The main area shows a user profile for Megan Bowen and a sidebar with 'Sample queries' and 'History'. At the top right, there are dropdowns for 'GET', 'v1.0', and a URL field containing 'https://graph.microsoft.com/v1.0/users/IsaiahL@MSDx927799.OnMicrosoft.com'. Below these are buttons for 'Request body', 'Request headers', 'Modify permissions (Preview)', and 'Access token'. A red arrow points to the 'Modify permissions (Preview)' button. The main content area displays a table titled 'Permissions (14)'. The table has columns for 'Permission', 'Description', 'Admin consent required', and 'Status'. All four rows in the table have their 'Status' button labeled 'Consent'. A second red arrow points to the 'Status' column header.

Figure 17.16: Grant permissions to user

After relevant permissions are in place, we can go ahead and query data using Microsoft Graph Explorer. Let's go ahead and start with a user query.

In this example, I would like to view the account details of the

`IsaiahL@MSDx927799.OnMicrosoft.com` user. To do that, I am going to use the `GET HTTP`

method and the

`https://graph.microsoft.com/v1.0/users/IsaiahL@rebeladmin.OnMicrosoft.com` query.

The screenshot shows the Microsoft Graph Explorer interface. The URL field at the top right contains the query 'https://graph.microsoft.com/v1.0/users/IsaiahL@MSDx927799.OnMicrosoft.com'. The rest of the interface is identical to Figure 17.16, showing the 'Modify permissions (Preview)' tab with the 'Status' column showing 'Consent' for all permissions.

Figure 17.17: User query

Then Microsoft Graph responds to the query with the following data:

```
{
  "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users/$entity",
  "businessPhones": [
    "+1 918 555 0101"
  ],
  "displayName": "Isaiah Langer",
  "givenName": "Isaiah",
  "jobTitle": "Sales Rep",
  "mail": "IsaiahL@rebeladmin.OnMicrosoft.com",
  "mobilePhone": null,
  "officeLocation": "20/1101",
  "preferredLanguage": null,
  "surname": "Langer",
  "userPrincipalName": "IsaiahL@rebeladmin.OnMicrosoft.com",
  "id": "0bb2ceaa-244f-4ea9-aa71-"
}
```

Figure 17.18: Output of user query

According to the above data, the user has the jobTitle attribute value as Sales Rep. This user has been promoted recently and I need to change the title to Sales Manager. To do that, we need to use the PATCH HTTP method:

```
PATCH
https://graph.microsoft.com/v1.0/users/IsaiahL@rebeladmin.OnMicrosoft.com
```

Before I run the query under the request body, I define the new values I need to apply:

```
{
  "jobTitle": "Sales Manager"
}
```

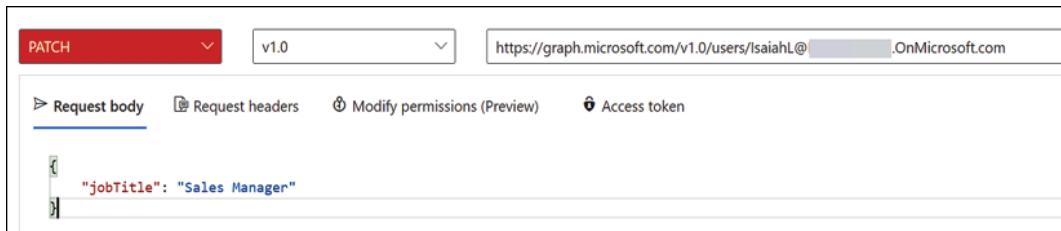


Figure 17.19: Update attribute value

After I run the query successfully, I can see the relevant attribute has a new value.

The screenshot shows the Microsoft Graph Explorer interface with the 'Response preview' tab selected. The JSON response for a user object includes several fields, with the 'jobTitle' field being highlighted by a red box. The JSON data is as follows:

```

{
    "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#users/$entity",
    "businessPhones": [
        "+1 918 555 0101"
    ],
    "displayName": "Isaiah Langer",
    "givenName": "Isaiah",
    "jobTitle": "Sales Manager", // This field is highlighted with a red box
    "mail": "IsaiahL@REDACTED.onmicrosoft.com",
    "mobilePhone": null,
    "officeLocation": "20/1101",
    "preferredLanguage": null,
    "surname": "Langer",
    "userPrincipalName": "IsaiahL@REDACTED.onmicrosoft.com",
    "id": "0bb2ceaa-244f-4ea9-aa71-910801f5bc75"
}

```

Figure 17.20: Confirm new attribute value

As the next step, I would like to list my domains under Azure AD. To do that, I can use:

```
GET https://graph.microsoft.com/v1.0/domains
```

It returns the details about all the domains under the current subscription. If we need to view data related to a particular domain, we can do it by using:

```
GET
https://graph.microsoft.com/v1.0/domains/rebeladmin.OnMicrosoft.com
```

In the preceding command, `rebeladmin.OnMicrosoft.com` is the FQDN.

The screenshot shows the Microsoft Graph Explorer interface with the 'Response preview' tab selected. The JSON response for a domain object includes various properties, such as authentication type, supported services, and password validity periods. The JSON data is as follows:

```

{
    "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#domains/$entity",
    "authenticationType": "Managed",
    "availabilityStatus": null,
    "id": "REDACTED.onmicrosoft.com",
    "isAdminManaged": true,
    "isDefault": true,
    "isInitial": true,
    "isRoot": true,
    "isVerified": true,
    "supportedServices": [
        "Email",
        "OfficeCommunicationsOnline"
    ],
    "state": null,
    "passwordValidityPeriodInDays": 2147483647,
    "passwordNotificationWindowInDays": 14
}

```

Figure 17.21: Domain information

As we can see, it provides rich data about the domain. Microsoft Graph Explorer can also be used to create, modify, and delete data by using the `HTTP` methods `POST` and `DELETE`. As an example, if the domain name is not verified, I can force the verification of the domain by using:

```
POST  
https://graph.microsoft.com/v1.0/domains/rebeladmin.onmicrosoft.com/ve  
rify
```

Next, I would like to list all the groups in Azure AD. For that, I can use:

```
GET https://graph.microsoft.com/v1.0/groups
```

The above command lists down all the groups in the directory with attribute values, but I am more interested in finding out the id and displayName values of the groups.

So, I am going to modify the query and only select id and displayName values:

```
GET https://graph.microsoft.com/v1.0/groups?$select=id,displayName
```

The screenshot shows the Microsoft Graph Explorer interface with the 'Response preview' tab selected. The JSON response lists several groups with their IDs and display names. The data is as follows:

```
{  
    "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#groups(id,displayName)",  
    "value": [  
        {  
            "id": "08a064c7-b9bb-4e4b-9885-2b2ebb2d0331",  
            "displayName": "Azure ATP msdx927799 Administrators"  
        },  
        {  
            "id": "0af59f8b-ce26-449c-ad87-fdd2f0fa63c2",  
            "displayName": "Sales Best Practices"  
        },  
        {  
            "id": "1015e0a0-25e1-4415-80c1-c41b7646b3c3",  
            "displayName": "Sales and Marketing"  
        },  
        {  
            "id": "12ac9125-d589-4b10-9cea-6756ed672349",  
            "displayName": "ssg-Contoso Ambassadors"  
        },  
        {  
            "id": "12bf1d10-3616-4ab5-8ad7-9990f086e91b",  
            "displayName": "sg-Executive"  
        },  
        {  
            "id": "28f3c738-35af-455b-8a61-0cf2bb5224d1",  
            "displayName": "Digital Initiative Public Relations"  
        }  
    ]  
}
```

Figure 17.22: List groups

But this is still a long list. I want to search for groups that start with the letters sg. To do that, we can use:

```
GET  
https://graph.microsoft.com/v1.0/groups?$filter=startswith(displayName  
, 'sg') &$select=id,displayName
```

From the group list, I am more interested in the sg-Sales and Marketing. Let's take note of the group id value for future use.

```

    ],
    [
      {
        "id": "c9311706-af26-43f4-bb3b-d18db4d63f18",
        "displayName": "sg-Operations"
      },
      {
        "id": "c57ea282-5515-40d2-9b28-1cb38256f7f6",
        "displayName": "sg-Retail"
      },
      [
        {
          "id": "eb2f21de-fae1-43cb-8594-8a39cb33de9d",
          "displayName": "sg-Sales and Marketing"
        }
      ]
    ]
  ]
}

```

Figure 17.23: List of groups contain "sg"

As the next step, I would like to get a list of members of sg-Sales and Marketing. We can do this by using:

```
GET https://graph.microsoft.com/v1.0/groups/{eb2f21de-fae1-43cb-8594-8a39cb33de9d}/members?$count=true
```

In the above command, `eb2f21de-fae1-43cb-8594-8a39cb33de9d` is the group ID of the sg-Sales and Marketing group.

```

{
  "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#directoryObjects",
  "@odata.count": 8,
  "value": [
    {
      "@odata.type": "#microsoft.graph.user",
      "id": "e991341c-53b7-48dd-8ad3-35aa795b8a16",
      "businessPhones": [
        "+1 425 555 0109"
      ],
      "displayName": "Adele Vance",
      "givenName": "Adele",
      "jobTitle": "Retail Manager",
      "mail": "AdeleV@contoso.onmicrosoft.com",
      "mobilePhone": null,
      "officeLocation": "18/2111",
      "preferredLanguage": null,
      "surname": "Vance",
      "userPrincipalName": "AdeleV@contoso.onmicrosoft.com"
    },
    {
      ...
    }
  ]
}

```

Figure 17.24: List members of a group

As we can see above, the query has a response with a list of all the users. In response, the `@odata.count` value represents the number of members in the group.

I have a new user called Megan with the object ID `086ba971-ecc2-49a5-a063-6e2d4fba9e9b`. I would like to add this user to the sg-Sales and Marketing group. This task includes two steps.

As the first step, we need to paste the following command in the query window:

```
POST https://graph.microsoft.com/v1.0/groups/{eb2f21de-fae1-43cb-8594-8a39cb33de9d}/members/$ref
```

In the above, `eb2f21de-fae1-43cb-8594-8a39cb33de9d` is the group ID value.

Then, under the Request body section, I use the following command:

```
{  
    "@odata.id":  
    "https://graph.microsoft.com/v1.0/directoryObjects/{086ba971-ecc2-49a5-a063-6e2d4fba9e9b}"  
}
```

This is the reference for the new member object ID. In the above, `086ba971-ecc2-49a5-a063-6e2d4fba9e9b` is the object ID of the user Megan.

After that, we can run the query. If it is successful, we should get a `No Content - 204 response`.

After that, I rerun the command to list down members of sg-Sales and Marketing and now I can see the new user as a member.

The screenshot shows the Microsoft Graph API Response preview interface. At the top, it says "OK - 200 - 1260ms". Below that is a note: "When you use Microsoft Graph APIs, you agree to the Microsoft APIs Terms of Use. View the Microsoft Privacy Statement." The main area shows a JSON response. The response includes an "@odata.context" key pointing to the metadata endpoint, an "@odata.count" key with the value "9", and a "value" key which is an array of user objects. One specific user object is highlighted with a red box, showing its properties: id, displayName, givenName, jobTitle, mail, mobilePhone, officeLocation, preferredLanguage, surname, and userPrincipalName. The "id" field is explicitly highlighted with a red box.

Figure 17.25: Add user to a group

We also can delete a member from a group by using the `DELETE HTTP` method. If we need to remove Megan from the group, we can use:

```
DELETE https://graph.microsoft.com/v1.0/groups/{eb2f21de-fae1-43cb-8594-8a39cb33de9d}/members/{086ba971-ecc2-49a5-a063-6e2d4fba9e9b}/$ref
```

In the preceding command, `eb2f21de-fae1-43cb-8594-8a39cb33de9d` is the object ID of the group and `086ba971-ecc2-49a5-a063-6e2d4fba9e9b` is the object ID of the user Megan.

In this section of the chapter, I have used several examples to show how we can use Microsoft Graph with Azure AD. This is a vast topic but I believe, now, you have an idea of how Microsoft Graph works. For more information, please visit <https://bit.ly/3HS0MJq>.

Summary

PowerShell has become the most powerful script language for Windows systems. PowerShell is very useful for systems management but is also an incredibly powerful tool for managing Active Directory infrastructures. Throughout the book, I have used PowerShell for Active Directory configuration and management.

Furthermore, I have shared different commands and scripts that can be used to manage an Active Directory environment efficiently.

Toward the end of the chapter, you learned how to manage Azure AD using the Azure Active Directory PowerShell for Graph module. We also looked into Microsoft Graph and learned how to use it to manage Azure AD. In the next chapter, we will look at Azure AD closely and learn how to manage identities in a hybrid environment.

Chapter 18

Links

Over the years, I have written many articles about Azure AD-related security features. I have listed some of those for reference:

- Step-by-Step Guide to Azure AD Privileged Identity Management – Part 1: <https://bit.ly/3rkJ1Np>
- Step-by-Step Guide to Azure AD Privileged Identity Management – Part 2: <https://bit.ly/3oY6QHW>
- Step-by-Step guide to setup temporally privilege access using Azure AD Privileged Identity Management: <https://bit.ly/3nKZgkm>
- Step-by-Step guide: Privileged access management in office 365: <https://bit.ly/2ZjMBf1>
- Step-by-Step Guide: Protect confidential data using Azure information protection: <https://bit.ly/30Ulwy2>
- Step-by-Step Guide: Automatic Data Classification via Azure Information Protection: <https://bit.ly/3cHejVY>
- Step-by-Step Guide: On-premise Data Protection via Azure Information Protection Scanner: <https://bit.ly/3DNPO5f>
- Step-by-Step Guide: How to protect confidential emails using Azure information protection?: <https://bit.ly/3FH7sID>
- Step-by-Step Guide: How to track shared documents using Azure information Protection?: <https://bit.ly/3xq3MYP>
- Step-by-Step Guide to Azure AD Password-less Authentication (public-preview): <https://bit.ly/3l7qiAV>
- Step-by-Step Guide: Using Microsoft Authenticator app (Public preview) to reset Azure AD user password: <https://bit.ly/3cO16KU>
- Azure AD Self-Service password reset for Windows 7/8.1 Devices: <https://bit.ly/32ukbjE>
- Conditional Access Policies with Azure Active Directory: <https://bit.ly/3l4Tb0v>
- Conditional Access with Azure AD B2B: <https://bit.ly/3l4AtpX>
- Step-by-Step guide to control data access using Azure cloud app security (based on content type): <https://bit.ly/3FltlqX>

- Step-by-Step guide to manage Impossible travel activity alert using Azure cloud app security: <https://bit.ly/3r4p792>
- Step-by-Step guide to block data download using Azure Cloud App security: <https://bit.ly/3r2PCeX>
- More info about Microsoft's Cloud Adoption Framework (CAF) is available on bit.ly/3nLQW3W.
- For more details on cost of different Azure AD versions, visit bit.ly/3cGtNtm.
- Azure AD password-less authentication: bit.ly/3HJ6g9z
- Hashes stored in AD are in the Message Digest 4 (MD4) algorithm format (bit.ly/3l4Tvfj).
- Password-Based Key Derivation Function 2 (PBKDF2) (bit.ly/3cJ5JpW).
- Valid Azure subscription: bit.ly/3oV95Ma
- More information about ports can be found at bit.ly/3r5zJo5. The service URL and IP range information can be found at <https://bit.ly/3nManck>.
- Azure portal as Global Administrator (<https://bit.ly/30XzSig>).
- There are three levels of subscription for the service and you can find more details about these on <https://bit.ly/3nLcc4Y>.
- Download the latest version of Azure AD Connect from <https://bit.ly/3l6JZZr>.

Figures

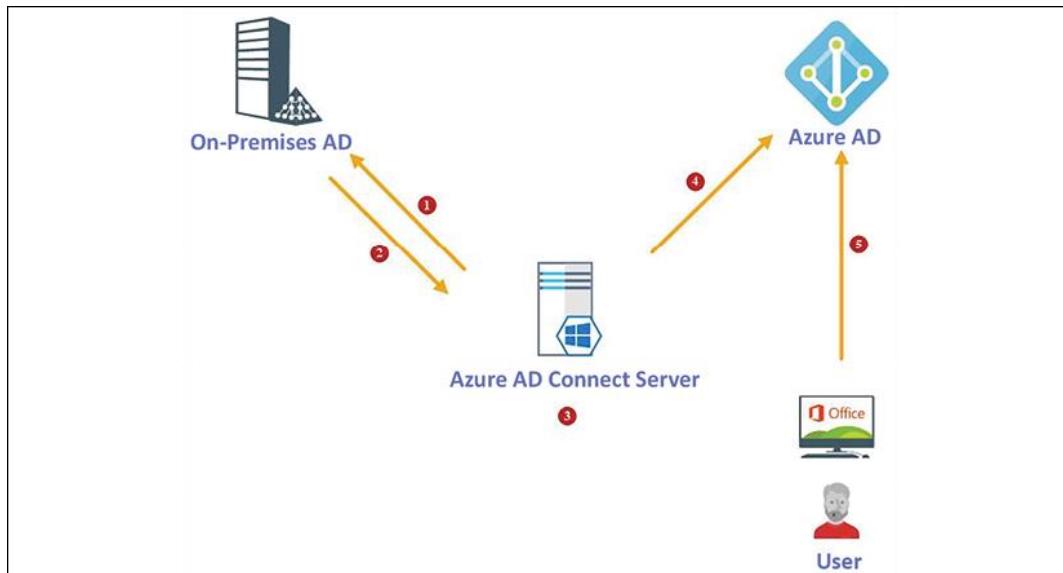


Figure 18.1: Password hash synchronization process

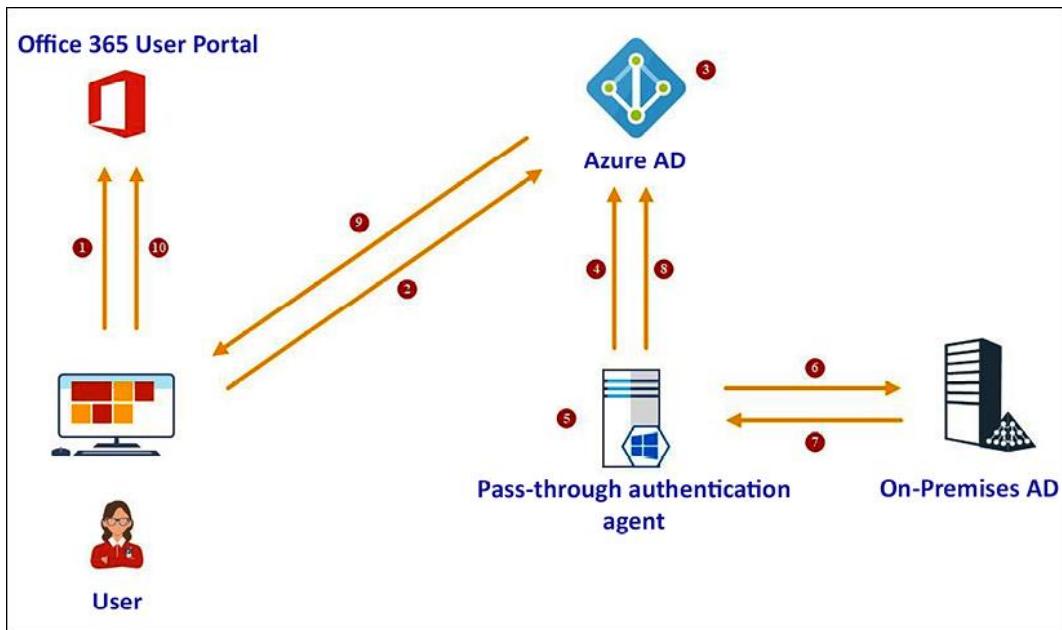


Figure 18.2: Pass-through authentication flow

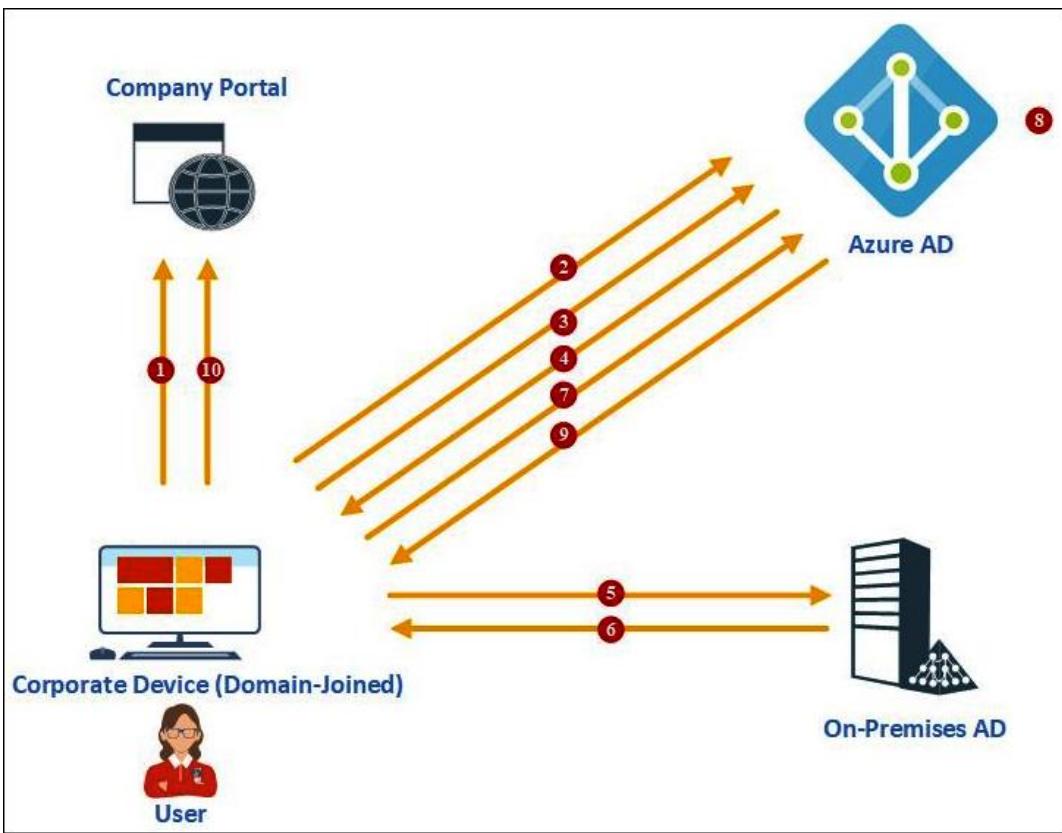


Figure 18.3: Seamless-SSO flow

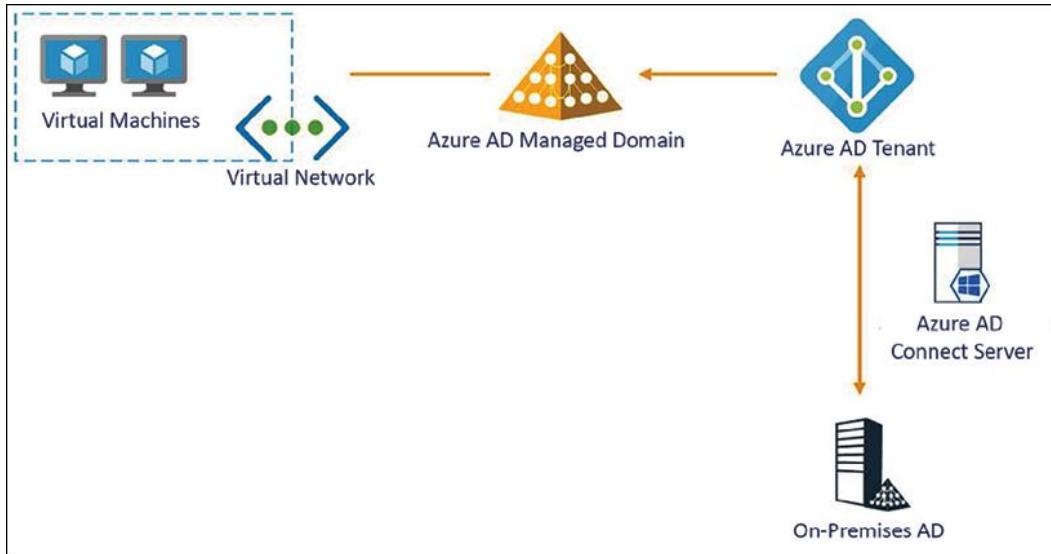


Figure 18.4: Azure AD hybrid topology with Azure AD managed domain

Contoso | Azure AD Connect

PROVISION FROM ACTIVE DIRECTORY

Azure AD cloud sync

This feature allows you to manage sync configurations from the cloud, in addition to syncing Active Directory users and groups from disconnected forests.

Manage Azure AD cloud sync 2

Not Installed	Download Azure AD Connect
Last Sync	Sync has never run
Password Hash Sync	Disabled

USER SIGN-IN

Federation	Disabled	0 domains
Seamless single sign-on	Disabled	0 domains
Pass-through authentication	Disabled	0 agents

Figure 18.5: Manage Azure AD cloud sync

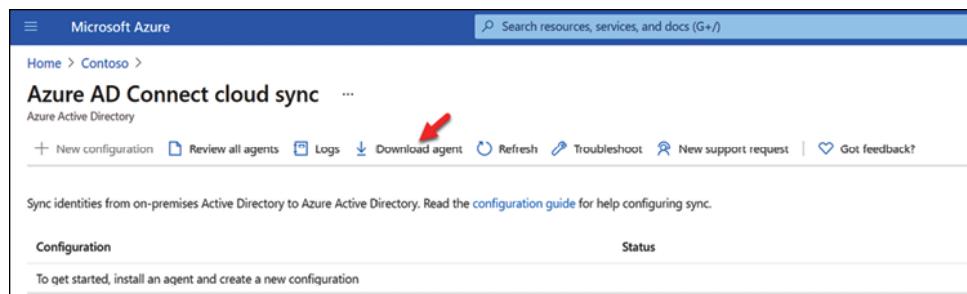


Figure 18.6: Download Azure AD cloud sync agent

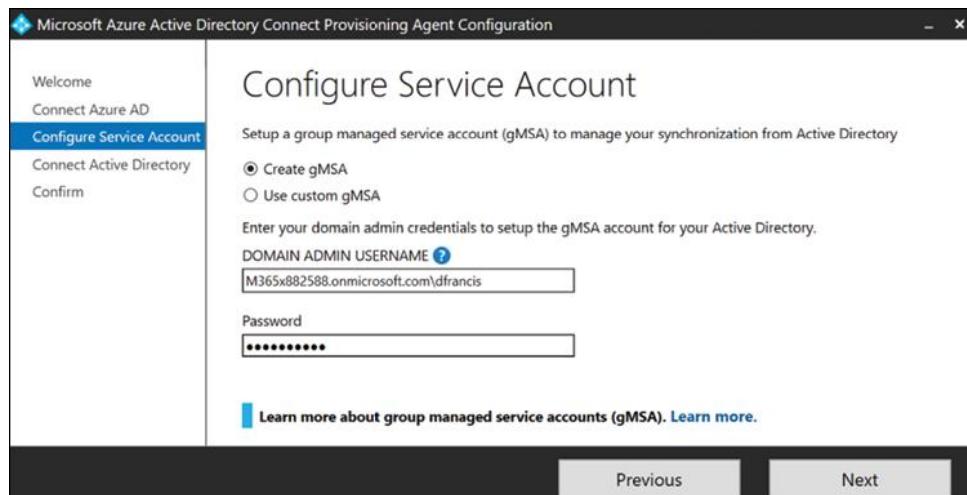


Figure 18.7: Create a gMSA account

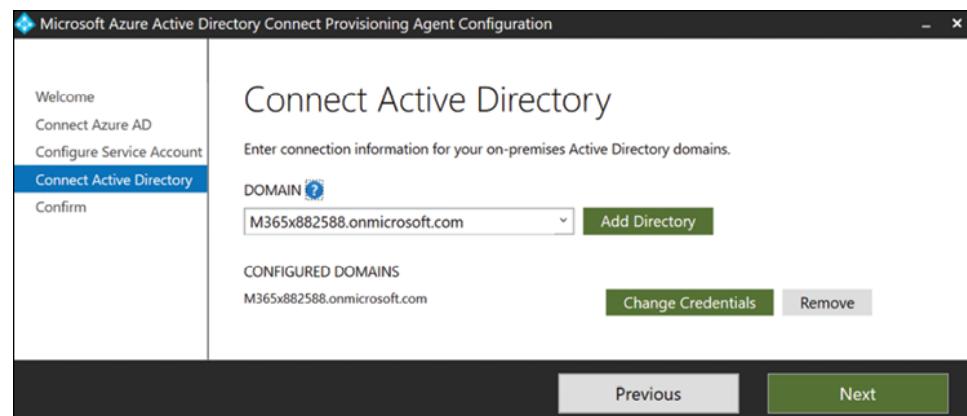


Figure 18.8: Connect to local AD

The screenshot shows the Azure AD Connect cloud sync configuration page. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below it, the path 'Home > Contoso > Azure AD Connect cloud sync' is shown. A red arrow points to the 'Review all agents' button in the top navigation bar, which is highlighted in blue. The main content area has sections for 'Configuration' and 'Status'. A message at the bottom says 'To get started, install an agent and create a new configuration'.

Figure 18.9: Review all agents option

The screenshot shows the 'On-premises provisioning agents' status page. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below it, the path 'Home > Contoso > Azure AD Connect cloud sync > On-premises provisioning agents' is shown. A red arrow points to the 'Download on-premises agent' button. The main content area displays a table with one row, showing the machine name 'DC01', external IP '13.82.69.150', and status 'active'.

Figure 18.10: Health status of provisioning agents

The screenshot shows the 'Azure AD Connect cloud sync' configuration page. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below it, the path 'Home > Contoso > Azure AD Connect cloud sync' is shown. A red arrow points to the 'New configuration' button in the top navigation bar. The main content area has sections for 'Configuration' and 'Status'. A message at the bottom says 'To get started, install an agent and create a new configuration'.

Figure 18.11: Set up new Azure AD cloud sync configuration

The screenshot shows the 'New cloud sync configuration' setup page. At the top, there's a navigation bar with 'Microsoft Azure' and a search bar. Below it, the path 'Home > Contoso > Azure AD Connect cloud sync > New cloud sync configuration' is shown. A red arrow points to the 'Create' button at the bottom of the page. The main content area includes a note about creating a configuration, a dropdown for the Active Directory domain ('M365x882588.onmicrosoft.com'), a checkbox for 'Enable password hash sync', and a 'Next steps' section with a note about advanced settings.

Figure 18.12: Azure AD cloud sync configuration – domain settings

The screenshot shows the 'Edit cloud sync configuration' page in the Azure Active Directory portal. The top navigation bar includes 'Home > Contoso > Azure AD Connect cloud sync >'. Below the title, there are buttons for 'Save' (with a red arrow pointing to it), 'Restart sync', and 'Delete'. The main area is titled 'Configure' and contains a note to read the 'configuration guide'. A numbered list from 1 to 5 guides the user through the configuration process:

- 1 Scope**
Active Directory domain: M365x882588.onmicrosoft.com
Scope users: All users in scope
Click to edit scoping filters
- 2 Manage attributes**
Sync password hashes: Enable
Map attributes: Click to edit mappings
- 3 Validate (recommended)**
Verify that sync is working as expected before enabling the configuration by testing with individual users. Quickly create, update, or disable the user's account in the target app based on your configuration.
Provision a user
- 4 Settings**
Notification email: admin@M365x882588.onmicrosoft.com
Prevent accidental deletion:
Accidental delete threshold: 500
- 5 Deploy**
Enabling this will sync the users and groups that are in scope as identified by this configuration.
Enable Disable

Figure 18.13: Save Azure AD cloud sync configuration settings

Search users Add filters

807 users found

Name	User principal name	User type	Directory synced	Identity issuer	Company name
AP Abraham Pawlik	Abraham.Pawlik@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AM Adelgunde Möbius	Adelgunde.M_mbius@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AD Adelheid Demmer	Adelheid.Demmer@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AH Adolfine Hägele	Adolfine.H_gele@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AH Agata Hammer	Agata.Hammer@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AK Albert Klett	Albert.Klett@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AL Alex Lörh	Alex.L_lohr@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AR Alex Reimer	Alex.Reimer@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AW Alex Wilber	Alex.W@M365x882588.onmicrosoft.com	Member	No		
AW Alf Wirth	Alf.Wirth@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AB Alhard Baier	Alhard.Baier@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AS Alida Schirmer	Alida.Schirmer@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AD Allan Deyoung	Allan.D@M365x882588.onmicrosoft.com	Member	No		
AH Alma Hennemann	Alma.Hennemann@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AS Alma Schweikert	Alma.Schweikert@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AH Aloysius Heß	Aloysius.He_ss@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AR Arneke Rieß	Arneke.Rie_ss@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AH Ambrosius Huppertz	Ambrosius.Huppertz@M365x882...	Member	Yes		Rebeladmin Corp.
AA Amelie Aust	Amelie.Aust@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.
AS Amelie Spieker	Amelie.Spieker@M365x882588.onmicrosoft.com	Member	Yes		Rebeladmin Corp.

Load more

Figure 18.14: Sync status of user accounts

Create virtual network

Basics IP Addresses Security Tags Review + create

Azure Virtual Network (VNet) is the fundamental building block for your private network in Azure. VNet enables many types of Azure resources, such as Azure Virtual Machines (VM), to securely communicate with each other, the internet, and on-premises networks. VNet is similar to a traditional network that you'd operate in your own data center, but brings with it additional benefits of Azure's infrastructure such as scale, availability, and isolation. [Learn more about virtual network](#)

Project details

Subscription * REBELSS

Resource group * REBELBOOK

[Create new](#)

Instance details

Name * REBELVMNet

Region * (US) East US

[Review + create](#) < Previous Next : IP Addresses > Download a template for automation

Figure 18.15: Create a new virtual network

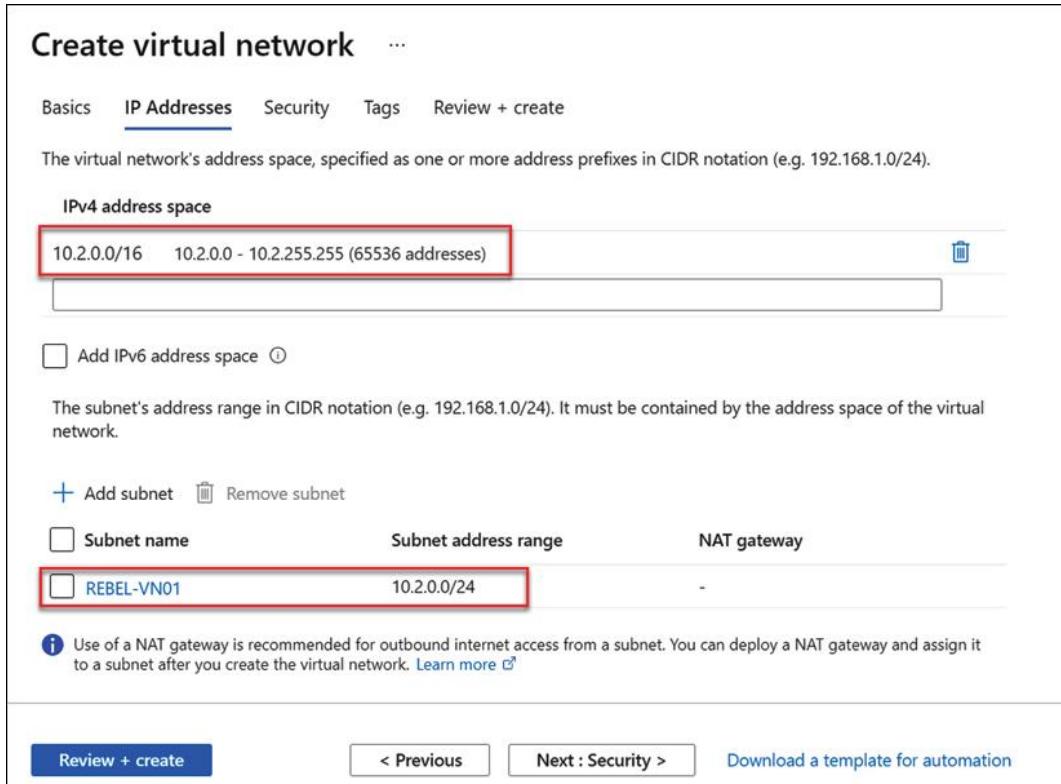


Figure 18.16: Virtual network subnet settings

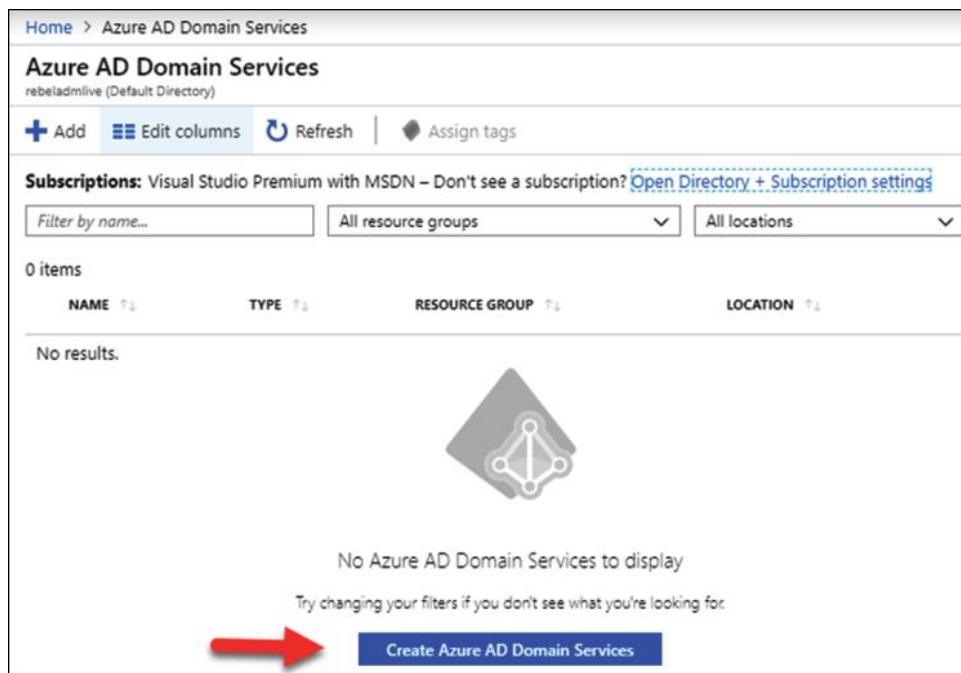


Figure 18.17: Initiating Azure AD DS setup

Create Azure AD Domain Services

* Basics * Networking Administration Synchronization Security Settings Tags Review + create

Azure AD Domain Services provides managed domain services such as domain join, group policy, LDAP, and Kerberos/NTLM authentication. You can use Azure AD Domain Services without needing to manage, patch, or service domain controllers in the cloud. For ease and simplicity, defaults have been specified to provide a one-click deployment. [Learn more](#)

Project details

When choosing the basic information needed for Azure AD Domain Services, keep in mind that the subscription, resource group, DNS domain name, and location cannot be changed after creation.

Subscription *	REBELSS
Resource group *	REBELBOOK
	Create new

Help me choose the subscription and resource group

DNS domain name *	rebeladmlive.onmicrosoft.com
-------------------	------------------------------

Help me choose the DNS name

Region *	(US) East US
SKU *	Enterprise

Help me choose a SKU

Forest type *	User	Resource
---------------	------	----------

Help me choose a forest type

Review + create Previous Next

Figure 18.18: Azure AD DS configuration

Home > Azure AD Domain Services >

Create Azure AD Domain Services

* Basics * Networking Administration Synchronization Security Settings Tags Review + create

Azure AD Domain Services uses a dedicated subnet within a virtual network to hold all of its resources. If using an existing network, ensure that the network configuration does not block the ports required for Azure AD Domain Services to run. [Learn more](#)

Virtual network *	REBELVMNet
	Create new

Help me choose the virtual network and address

Subnet *	REBEL-VN01 (10.2.0.0/24)
	Manage

Help me choose the subnet and NSG

Info A network security group will be automatically created and associated to the subnet to protect AAD Domain Services. The network security group will be configured according to [guidelines for configuring NSGs](#).

Review + create Previous Next

Figure 18.19: Azure AD DS – virtual network settings

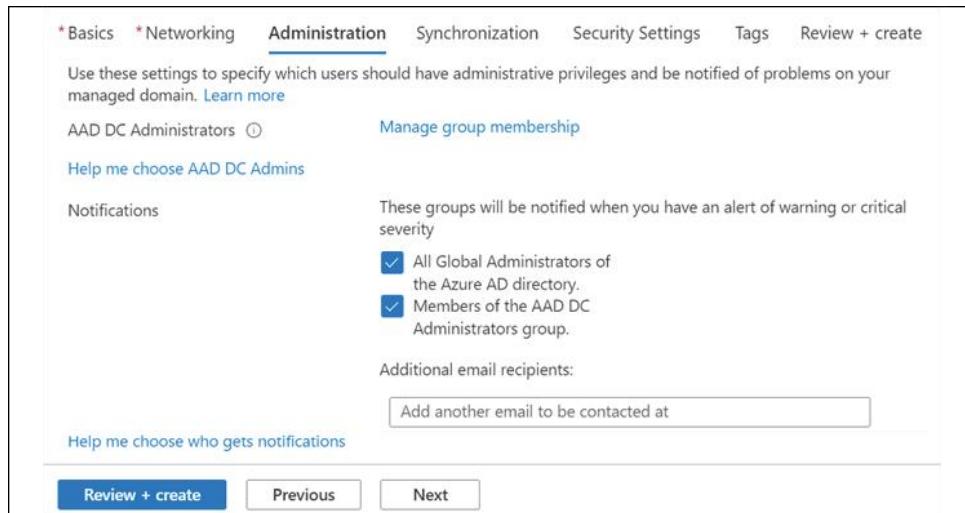


Figure 18.20: Azure AD DS – Domain Admin settings

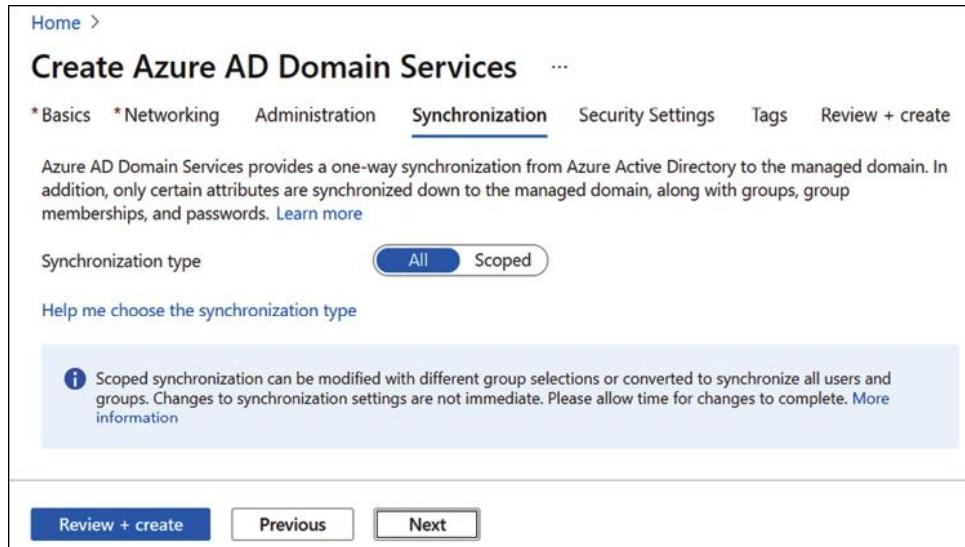


Figure 18.21: Azure AD DS – synchronization scope

Azure AD Domain Services				
Subscription: Visual Studio Premium with MSDN - Don't see a subscription? Open Directory + Subscription settings				
NAME	TYPE	RESOURCE GROUP	LOCATION	SUBSCRIPTION
rebeladmin.onmicrosoft.com	Azure AD Domain Services	REBELDC	East US	Visual Studio Premium with MSDN

Figure 18.22: Azure AD DS instance

Required configuration steps



Update DNS server settings for your virtual network

Update the DNS server settings for your virtual network to point to the IP addresses (10.2.0.4 and 10.2.0.5) where Azure AD Domain Services is available.

[More information](#)

[Configure](#)

Figure 18.23: Update DNS records

Home > rebeladmlive.onmicrosoft.com > REBELVMNet

REBELVMNet | DNS servers ...

Virtual network

Search (Ctrl+ /)

[Overview](#)

[Activity log](#)

[Access control \(IAM\)](#)

[Tags](#)

[Diagnose and solve problems](#)

Settings

[Address space](#)

[Connected devices](#)

[Subnets](#)

[DDoS protection](#)

[Firewall](#)

[Security](#)

[DNS servers](#)

[Peerings](#)

[Service endpoints](#)

[Private endpoints](#)

DNS servers

Default (Azure-provided)

Custom

IP Address

10.2.0.4

10.2.0.5

[Add DNS server](#)

Figure 18.24: Custom DNS servers

The screenshot shows the 'Users - All users' page in Azure Active Directory. On the left, there's a sidebar with options like 'All users', 'Deleted users', 'Password reset', 'User settings', 'Activity', 'Sign-ins', 'Audit logs', 'Troubleshooting + Support', 'Troubleshoot', and 'New support request'. The main area has a search bar and a table of users. A red arrow points to the 'New user' button at the top right.

Name	Show
DA DC Admin	
DF Disha Francis	
US user1	
US user2	
VD vdadmin	
VD vdcadmin	

Figure 18.25: Create new user

The screenshot shows the 'REBELADMIN - Azure AD Connect' page. The left sidebar includes 'Overview', 'Getting started', 'Manage' (with 'Users', 'Groups', 'Organizational relationships', 'Roles and administrators', 'Enterprise applications', 'Devices', 'App registrations', 'Identity Governance', 'Application proxy', 'Licenses', 'Azure AD Connect' which is selected and highlighted in blue), 'Custom domain names', and 'Mobility (MDM and MAM)'. The main area displays 'SYNC STATUS' (Not Installed, Last Sync, Password Hash Sync), 'USER SIGN-IN' (Federation, Seamless single sign-on, Pass-through authentication which is highlighted with a red box), 'ON-PREMISES APPLICATIONS' (looking to configure remote access for on-premises applications), and 'HEALTH AND ANALYTICS' (monitor your on-premises identity infrastructure and synchronization services in the cloud).

Figure 18.26: Pass-through authentication settings

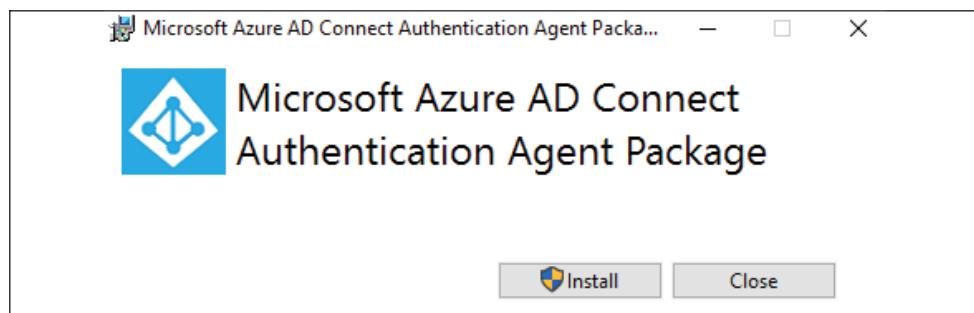


Figure 18.27: Start Azure AD Connect authentication agent installation

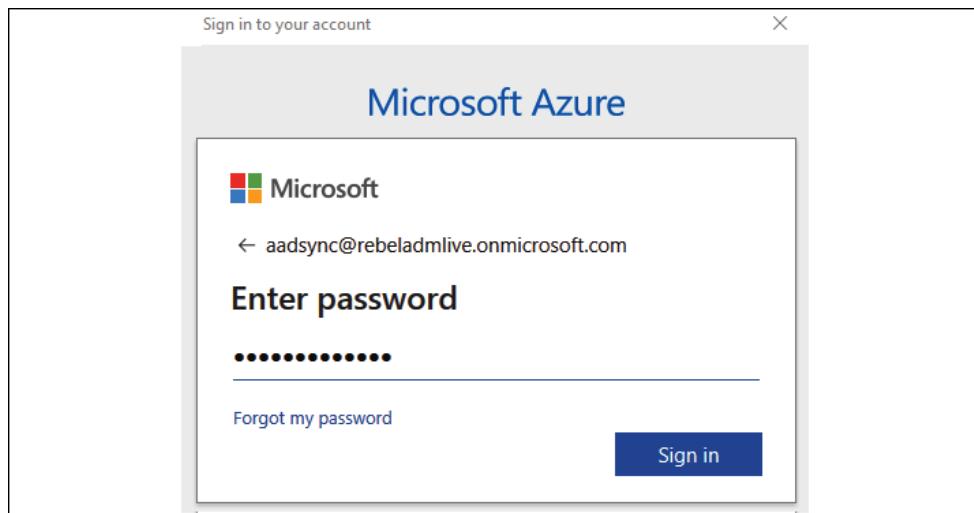


Figure 18.28: Azure AD authentication

A screenshot of the Azure portal under "Azure Active Directory" > "Pass-through authentication". The title is "Pass-through authentication". It shows a table with one row:

AUTHENTICATION AGENT	IP	STATUS	WARNINGS
▼ Default group for Pass-through Authentication RDC01.rebeladmlive.onmicrosoft.com	77.98.	Active	1

Figure 18.29: Pass-through authentication agent status

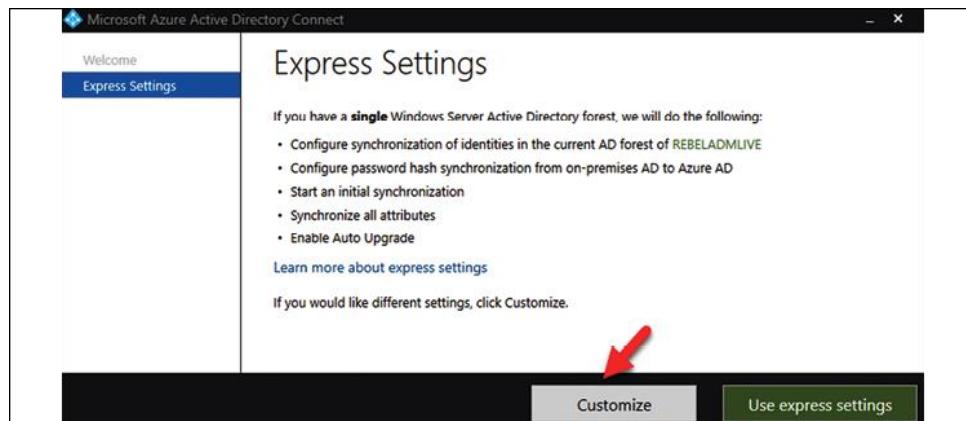


Figure 18.30: Azure AD Connect express setup

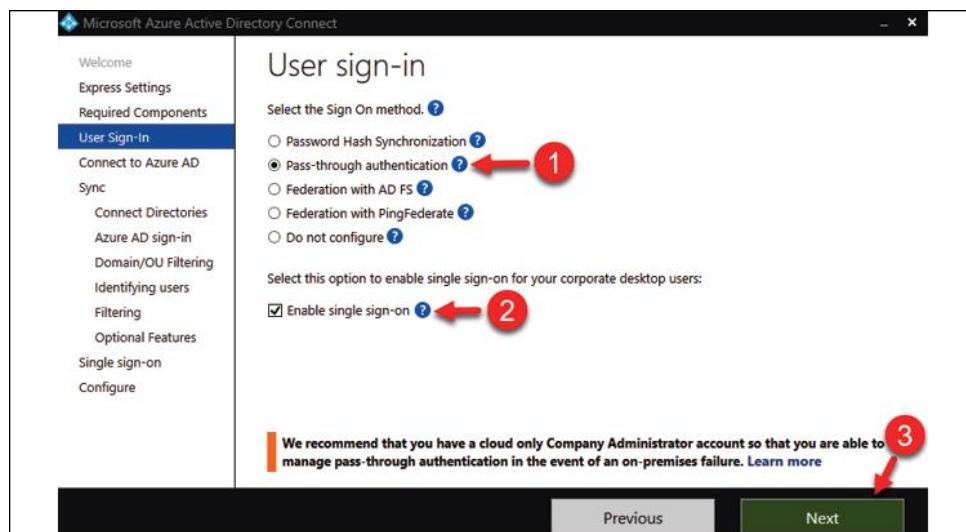


Figure 18.31: Azure AD Connect pass-through authentication setting

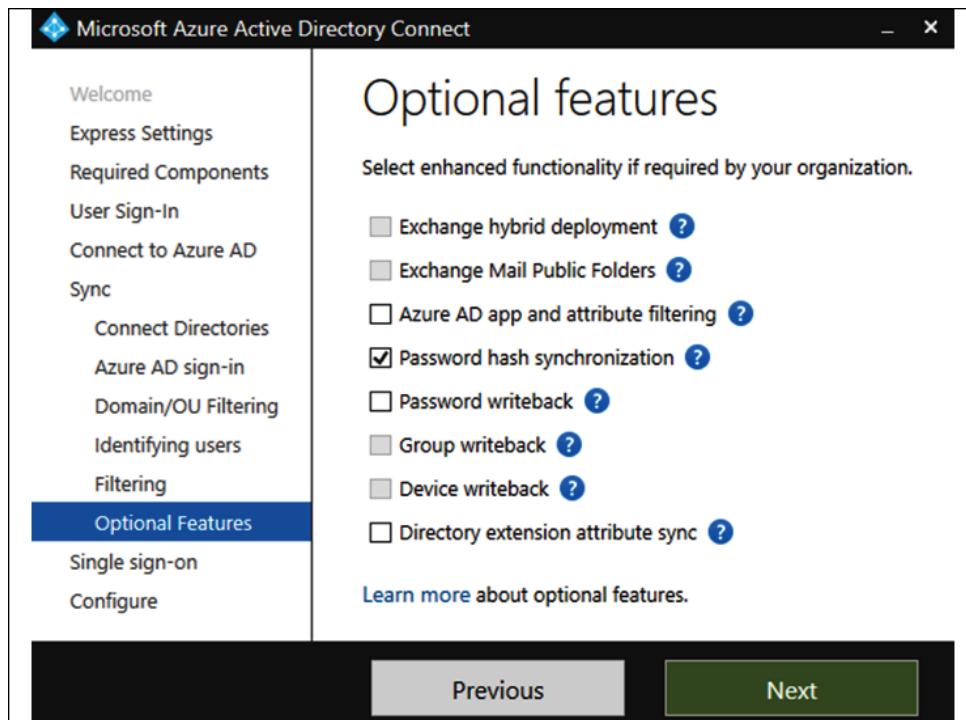


Figure 18.32: Azure AD Connect Optional features selection

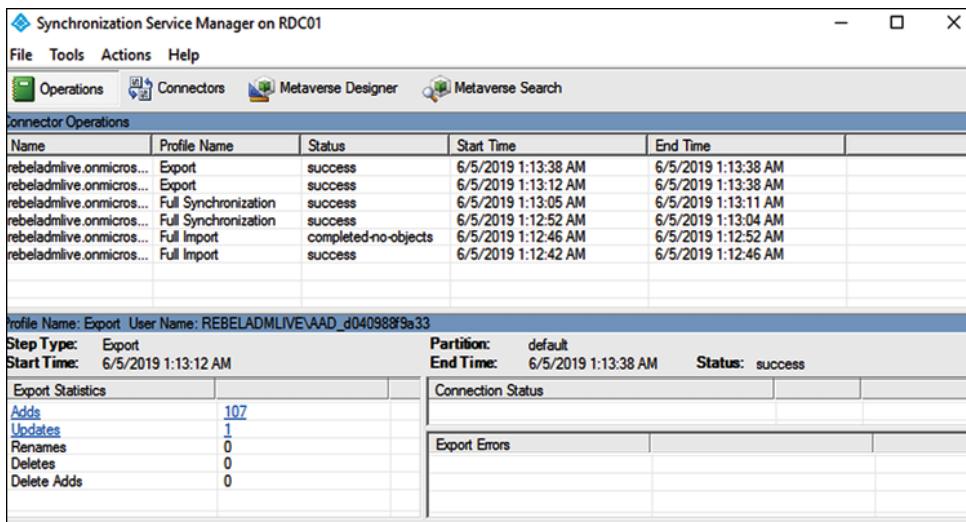


Figure 18.33: Azure AD Connect synchronization status

Name	UPN Name	User Type	Source
AADsync	AADsync@rebeladmin.onmicrosoft.com	Member	Azure Active Directory
Adelina Urtet	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Allard Spilmann	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Allard Witt	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Allert Wöhrle	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Anette Kappler	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Anemone Kellbich	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Analise Remane	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Analise Zelle	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Anely Grigor	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Arte Will	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Beatrix Gitsch	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Bettl Ehrlich	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Berwin Höller	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Brandolf Stau	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD
Carmen Exnor	III/IU7P5w@rebeladmin.onmicrosoft.com	Member	Windows Server AD

Figure 18.34: Azure AD user sync status

Name	Type	Description
rebeladmin.onmicrosoft.com - AAD	Windows Azure Active Directory (Mic...)	
rebeladmin.onmicrosoft.com	Active Directory Domain Services	

Figure 18.35: Connectors

Issued To	Issued By	Expiration Date	Intended Purposes	Friendly Name	Status
*rebeladmin.onmicrosoft.com	*rebeladmin.onmicrosoft.com	6/19/2021	Client Authentication	<None>	
Windows Azure CRP Certificate ...	Windows Azure CRP Certificate Ge...	6/19/2021	<All>	TenantEncryptionCert	

Figure 18.36: SSL certificate

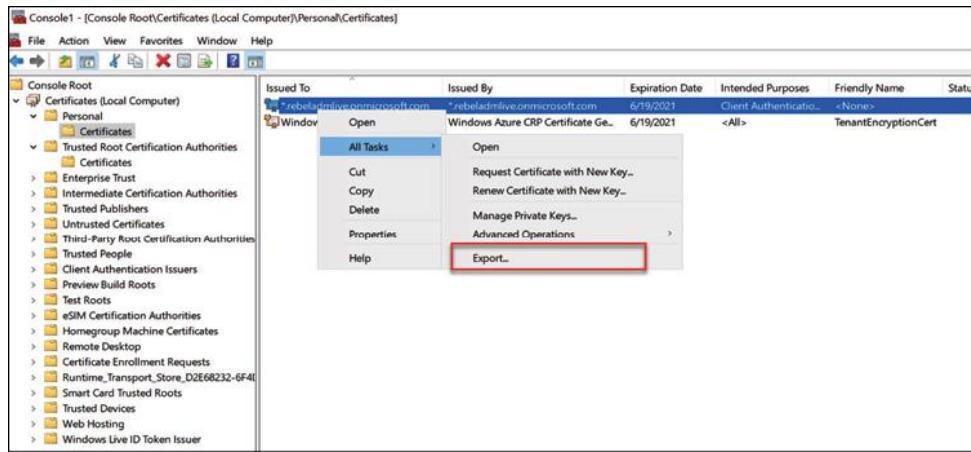


Figure 18.37: Export SSL certificate



Figure 18.38: Export private key

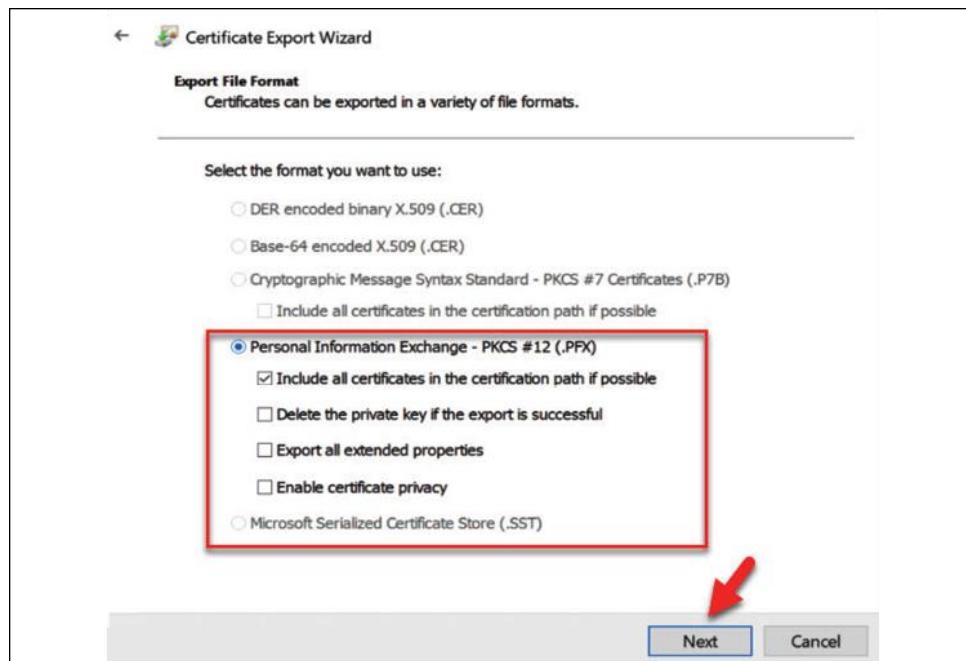


Figure 18.39: Export as PFX

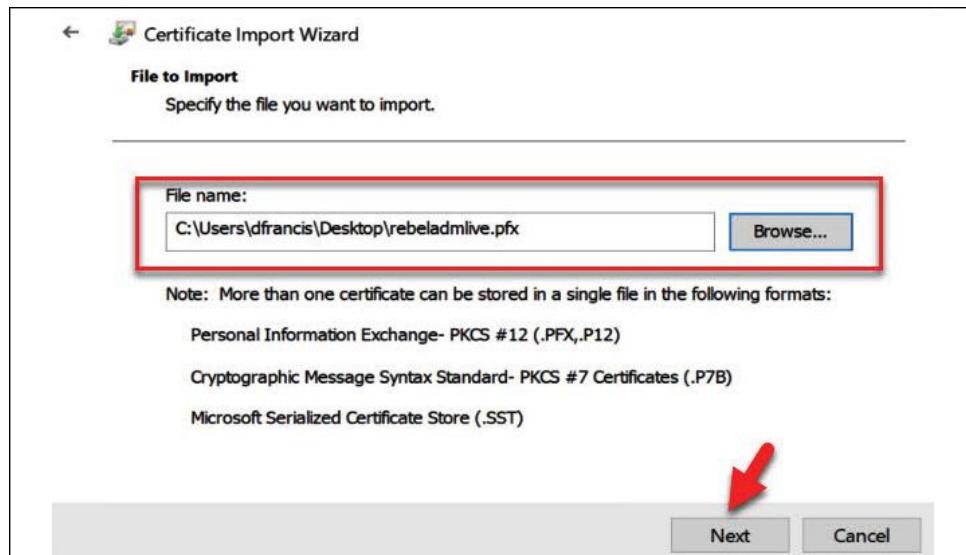


Figure 18.40: PFX file path

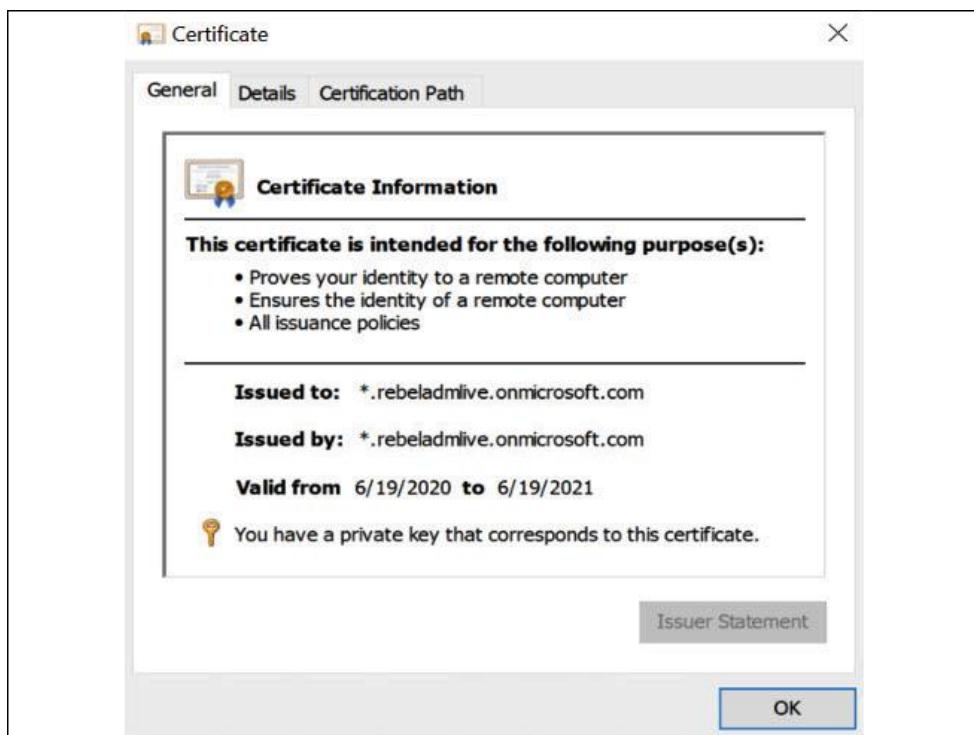


Figure 18.41: Certificate status

rebeladmlive.onmicrosoft.com
Azure AD Domain Services

Search (Ctrl+ /) < Delete

Overview

- Activity log
- Access control (IAM)

Settings

- Properties
- Secure LDAP**
- Synchronization
- Health
- Notification settings
- SKU

Monitoring

- Diagnostic settings
- Logs
- Workbooks

Support + troubleshooting

- Troubleshoot
- New support request

Azure AD Domain Services SKUs

Choose Azure AD Domain Services SKU for your organization

Azure AD Domain Services is available in multiple service tiers, known as SKUs. These SKUs provide predictable pricing, varying performance levels, and selectable enterprise and premium features.

[More information](#)

Choose SKU

Required configuration steps

Enable Azure AD Domain Services password hash synchronization

Users cannot bind using secure LDAP or sign in to the managed domain, until you enable password hash synchronization to Azure AD Domain Services. Follow the instructions below, depending on the type of users in your Azure AD directory. Complete both sets of instructions if you have a mix of cloud-only and synced user accounts in your Azure AD directory.

- Instructions for cloud-only user accounts
- Instructions for synced user accounts

Figure 18.42: Secure LDAP feature

rebeladmlive.onmicrosoft.com | Secure LDAP

Secure LDAP: Enabled Disabled

Thumbprint: Not available

Allow secure LDAP access over the internet: Disable Enable

Upload a PPK file containing the certificate to be used for secure LDAP access to this managed domain:

PPK file with secure LDAP certificate: rebeladmin.ppk

Password to decrypt PPK file:

Your subnet is protected by network security group asd1bg. To give user access to secure LDAP endpoint, please ensure 'Allow' rule on port 636 is configured with proper IP ranges on the network security group.

Users cannot bind using secure LDAP or sign in to the managed domain, until you enable password hash synchronization to Azure AD Domain Services. Follow the instructions below, depending on the type of users in your Azure AD directory. Complete both sets of instructions if you have a mix of cloud-only and synced user accounts.

- Instructions for cloud-only user accounts
- Instructions for synced user accounts

Figure 18.43: Configure secure LDAP

Home >

rebeladmlive.onmicrosoft.com | Properties ✖

Azure AD Domain Services

«

① Overview DNS domain name
rebeladmlive.onmicrosoft.com

⌚ Activity log Location
East US

🔍 Access control (IAM)

Settings

在家里 Properties SKU
Standard

🔒 Secure LDAP Forest type
User

⌚ Synchronization Available in virtual network/subnet
aadds-vnet/aadds-subnet

❤️ Health

✉️ Notification settings Network security group associated with subnet
aadds-nsg

⌚ SKU IP address on virtual network
10.0.0.5 10.0.0.4

Monitoring

💡 Diagnostic settings Secure LDAP
Enabled

Logs

📊 Workbooks Secure LDAP certificate thumbprint

Support + troubleshooting

✖ Troubleshoot Secure LDAP certificate expires
Sat, 19 Jun 2021 21:01:12 GMT

✉️ New support request Secure LDAP external IP address
52.188.114.225

The screenshot shows the 'Properties' page for the Azure AD Domain Service 'rebeladmlive.onmicrosoft.com'. The left sidebar has sections for Overview, Activity log, Access control (IAM), Settings (Properties selected), Monitoring, and Support + troubleshooting. The main content area displays various service details. A red box highlights the 'Secure LDAP external IP address' field, which contains the value '52.188.114.225'.

Figure 18.44: Public IP details

Home >

rebeladmlive.onmicrosoft.com | Properties

Azure AD Domain Services

Search (Ctrl+ /) <>

Overview

Activity log

Access control (IAM)

Properties

Secure LDAP

Synchronization

Health

Notification settings

SKU

Diagnostic settings

Logs

Workbooks

Troubleshoot

New support request

DNS domain name
rebeladmlive.onmicrosoft.com

Location
East US

SKU
Standard

Forest type
User

Available in virtual network/subnet
[aadds-vnet/aadds-subnet](#)

Network security group associated with subnet
aadds-nsg

IP address on virtual network
10.0.0.5 10.0.0.4

Secure LDAP
Enabled

Secure LDAP certificate thumbprint
[REDACTED]

Secure LDAP certificate expires
Sat, 19 Jun 2021 21:01:12 GMT

Secure LDAP external IP address
52.188.114.225

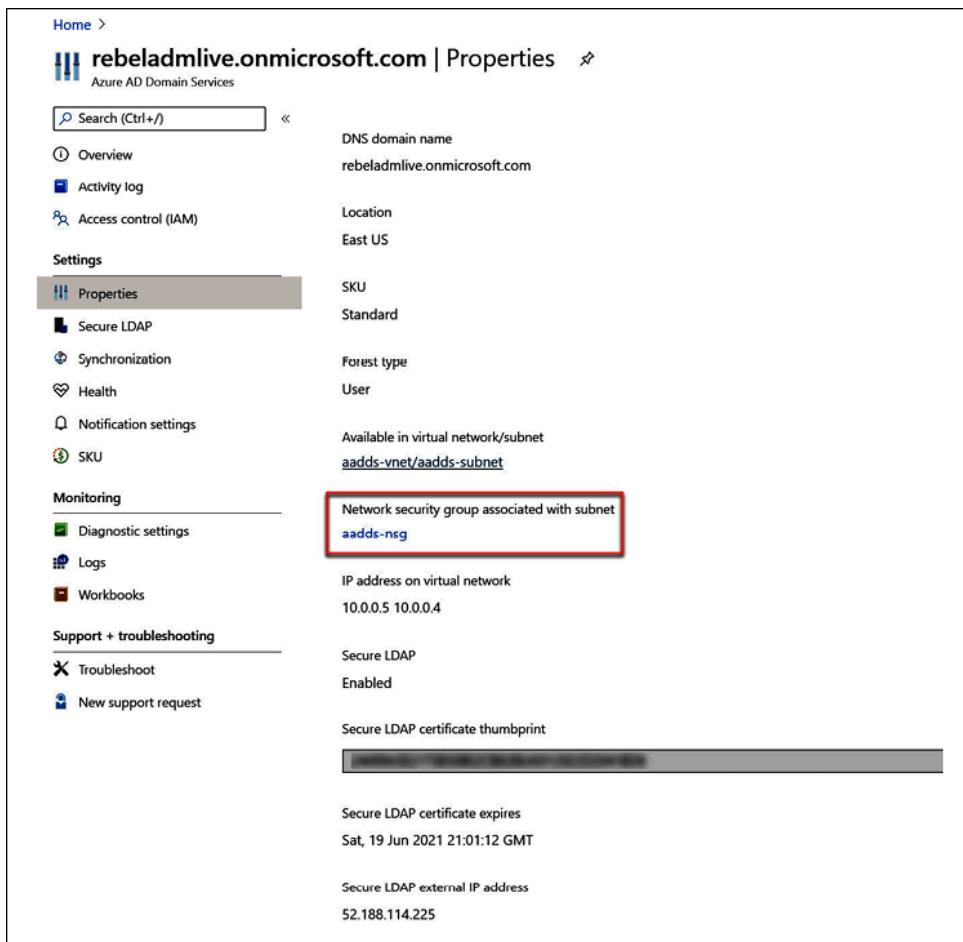
The screenshot shows the Azure AD Domain Services properties page for the domain 'rebeladmlive.onmicrosoft.com'. The 'Properties' tab is selected in the left navigation menu. In the main content area, under the 'Available in virtual network/subnet' section, there is a field labeled 'Network security group associated with subnet' containing the value 'aadds-nsg'. This value is highlighted with a red rectangular box. Other visible details include the IP address on the virtual network (10.0.0.5), the secure LDAP status (Enabled), and the secure LDAP certificate expiration date (Sat, 19 Jun 2021 21:01:12 GMT). The 'Secure LDAP certificate thumbprint' field is shown as a redacted gray bar.

Figure 18.45: NSG details

Add inbound security rule

aadds-nsg

Basic

Source * ⓘ
IP Addresses

Source IP addresses/CIDR ranges * ⓘ
52.136.127.107

Source port ranges * ⓘ
*

Destination * ⓘ
Any

Destination port ranges * ⓘ
636

Protocol *
Any TCP UDP ICMP

Action *
Allow Deny

Priority * ⓘ
311

Name *
secureLDAP

Description
Allow Secure LDAP

Add



Figure 18.46: Create a new inbound security rule

```

hosts - Notepad
File Edit Format View Help
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97      rhino.acme.com          # source server
#      38.25.63.10      x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#      127.0.0.1      localhost
#      ::1            localhost

52.188.114.225 ldaps.rebeladmlive.onmicrosoft.com

```

Figure 18.47: Host entry

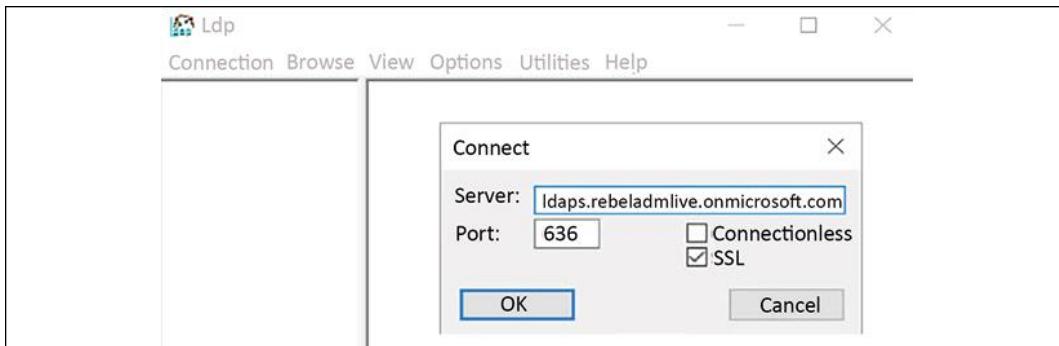


Figure 18.48: ldp.exe connection settings



Figure 18.49: Secure LDAP connection status

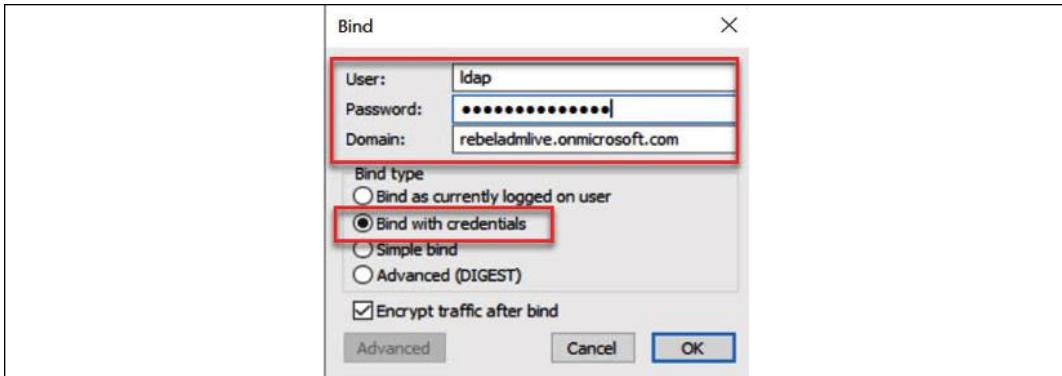


Figure 18.50: Bind with credentials

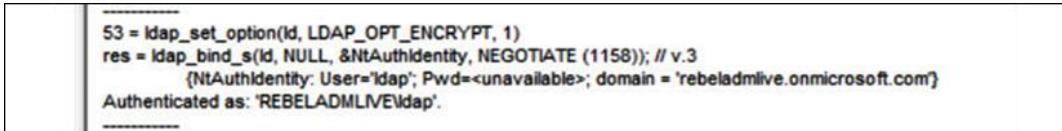


Figure 18.51 : Bind status



Figure 18.52: Browse data

 **rebeladmlive.onmicrosoft.com | Properties** ✖

Azure AD Domain Services

«

① Overview
② Activity log
③ Access control (IAM)

Settings

④ Properties (selected)
⑤ Secure LDAP
⑥ Synchronization
⑦ Replica sets (preview)
⑧ Health
⑨ Notification settings
⑩ SKU

Monitoring

⑪ Diagnostic settings
⑫ Logs
⑬ Workbooks

Support + troubleshooting

⑭ Troubleshoot
⑮ New support request

DNS domain name: rebeladmlive.onmicrosoft.com

Locations: East US

Forest type: User

Virtual Networks/Subnets
East US/REBELVN1/aadds-subnet

Network Security Groups: East US/aadds-nsg

IP addresses: East US/10.0.0.5 10.0.0.4

Secure LDAP: Disabled

Secure LDAP external IP addresses: East US/Not applicable

Synchronization: All

Admin group: AAD DC Administrators

Figure 18.53: Virtual network details

```

PS C:\WINDOWS\system32> Get-AzVirtualNetwork -Name REBELVNI -ResourceGroupName REBELRG1

Name          : REBELVNI
ResourceGroupName : REBELRG1
Location       : eastus
Id            : /subscriptions/REBELRG1/resourceGroups/REBELRG1/providers/Microsoft.Network/virtualNetworks/REBELVNI
Etag          : "060920b2-f2c4-4750-a683-2bf02c53e1fs"
ResourceGuid   : b92b07f0-bc0a-4be1-ac65-f6edd079b6e6
ProvisioningState : Succeeded
Tags          :
AddressSpace  : {
    "AddressPrefixes": [
        "10.0.0.0/16"
    ]
}
DhcpOptions   : {}
Subnets       : [
    {
        "Delegations": [],
        "Name": "vmsubnet",
        "Etag": "060920b2-f2c4-4750-a683-2bf02c53e1fs",
        "Id": "/subscriptions/REBELRG1/resourceGroups/REBELRG1/providers/Microsoft.Network/virtualNetworks/REBELVNI/subnets/vmsubnet",
        "AddressPrefix": [
            "10.0.2.0/24"
        ],
        "IpConfigurations": [],
        "ServiceAssociationLinks": [],
        "ResourceNavigationLinks": [],
        "ServiceEndpoints": [],
        "ServiceEndpointPolicies": [],
        "PrivateEndpoints": [],
        "ProvisioningState": "Succeeded",
        "PrivateEndpointNetworkPolicies": "Enabled",
        "PrivateLinkServiceNetworkPolicies": "Enabled"
    },
    {
        "Delegations": [],
        "Name": "aadds-subnet",
        "Etag": "060920b2-f2c4-4750-a683-2bf02c53e1fs",
        "Id": "/subscriptions/REBELRG1/resourceGroups/REBELRG1/providers/Microsoft.Network/virtualNetworks/REBELVNI/subnets/aadds-subnet",
        "AddressPrefix": [
            "10.0.0.0/24"
        ],
        "IpConfigurations": [
            {
                "Id": "/subscriptions/REBELRG1/resourceGroups/REBELRG1/providers/Microsoft.Network/networkInterfaces/aadds-a54e5dd88ed040ad99e0123dd9eff27d-nic/ipConfigurations/AK7YRUSX5DFUP-OIpcfg"
            },
            {
                "Id": "/subscriptions/REBELRG1/resourceGroups/REBELRG1/providers/Microsoft.Network/networkInterfaces/aadds-3af56a67fb2f4119b1d6a09c7a7dc4ee-nic/ipConfigurations/AY1216CZOX-HMJIpcfg"
            }
        ],
        "ServiceAssociationLinks": [],
        "ResourceNavigationLinks": [],
        "NetworkSecurityGroup": {
            "Id": "/subscriptions/REBELRG1/resourceGroups/REBELRG1/providers/Microsoft.Network/networkSecurityGroups/aadds-nsg"
        },
        "ServiceEndpoints": [],
        "ServiceEndpointPolicies": [],
        "PrivateEndpoints": [],
        "ProvisioningState": "Succeeded",
        "PrivateEndpointNetworkPolicies": "Enabled",
        "PrivateLinkServiceNetworkPolicies": "Enabled"
    }
]
VirtualNetworkPeering : []
EnableDdosProtection : false
DdosProtectionPlan   : null

```

Figure 18.54: Current configuration of the virtual network

```

PS C:\WINDOWS\system32> New-AzResourceGroup -Name REBELDRRG1 -Location "West US"

ResourceGroupName : REBELDRRG1
Location         : westus
ProvisioningState : Succeeded
Tags             :
ResourceId       : /subscriptions/REBELRG1/resourceGroups/REBELDRRG1

PS C:\WINDOWS\system32>

```

Figure 18.55: Create a resource group

```

PS C:\Windows\system32> New-AzureRmNetwork -Name REBELDRW1 -ResourceGroupName REBELDRRG1 -VirtualIpRange "10.1.0.16" -AddressPrefixes "10.1.0.0/16" -Subnet $drvsubnet
Name          : REBELDRW1
ResourceGroupName : REBELDRRG1
Location      : westus
Id            : /subscriptions//resourceGroups/REBELDRRG1/providers/Microsoft.Network/virtualNetworks/REBELDRW1
Tag          : "8349e9d-f70-4851-9781-7919a6c7d65"
ProvisioningState : Succeeded
Tags          :
AddressSpace  : [
    {
        "AddressPrefixes": [
            "10.1.0.0/16"
        ]
    }
]
Subnets       : [
    {
        "Delegations": [],
        "Name": "drvsubnet",
        "Etag": "0w/0ba9479-92d1-4d3d-adec-919a15ca434",
        "Properties": {
            "SubnetId": "/subscriptions//resourceGroups/REBELDRRG1/providers/Microsoft.Network/virtualNetworks/REBELDRW1/subnets/drvsubnet",
            "AddressPrefix": "10.1.0.0/24"
        },
        "IpConfigurations": [],
        "ServiceAssociationLinks": [],
        "ResourceNavigationLinks": [],
        "ServiceEndpoint": [],
        "PrivateEndpoint": [],
        "ProvisioningState": "Succeeded",
        "PrivateEndpointNetworkPolicies": "Enabled",
        "PrivateLinkServiceNetworkPolicies": "Enabled"
    }
]
VirtualNetworkPeerings : []
DisableDnsResolution : False
DnsZoneRegistration : null

```

Figure 18.56: Set up a new virtual network for a replica set

```

PS C:\Windows\system32> Add-AzureRmNetworkPeering -Name 00-001-0001-0000 -VirtualNetworkName REBELDRW1 -RemoteVirtualNetworkId 00-001-0001-0000
Name          : 00-001-0001-0001
Id            : /subscriptions//resourceGroups/00-001-0001/providers/Microsoft.Network/virtualNetworks/00-001-0001/peering/00-001-0001-0001
Tag          : "00-001-0001-0001-0001-peering"
ProvisioningState : Succeeded
VirtualNetworkName : REBELDRW1
RemoteVirtualNetworkId : /subscriptions//resourceGroups/00-001-0001/providers/Microsoft.Network/virtualNetworks/00-001-0001
VirtualNetworkPeerings : []
AllTrafficInboundNatRules : []
AllTrafficOutboundNatRules : []
AllTrafficEncryptionRequired : False
AllTrafficEncryptionTransit : False
AllowRemoteAccess : False
AllowVirtualNetworkPeering : True
remoteVirtualNetworkAddressSpace : [
    {
        "AddressPrefixes": [
            "10.0.0.0/16"
        ]
    }
]

```

Figure 18.57: Global VNet peering

```

PS C:\Windows\system32> Add-AzureRmNetworkPeering -Name 00-001-0001-0001 -VirtualNetworkName REBELDRW1 -RemoteVirtualNetworkId 00-001-0001
Name          : 00-001-0001-0001
Id            : /subscriptions//resourceGroups/00-001-0001/providers/Microsoft.Network/virtualNetworks/00-001-0001/peering/00-001-0001-0001
Tag          : "00-001-0001-0001-0001-peering"
ProvisioningState : Succeeded
VirtualNetworkName : REBELDRW1
RemoteVirtualNetworkId : /subscriptions//resourceGroups/00-001-0001/providers/Microsoft.Network/virtualNetworks/00-001-0001
VirtualNetworkPeerings : []
AllTrafficInboundNatRules : []
AllTrafficOutboundNatRules : []
AllTrafficEncryptionRequired : False
AllTrafficEncryptionTransit : False
AllowRemoteAccess : False
AllowVirtualNetworkPeering : True
remoteVirtualNetworkAddressSpace : [
    {
        "AddressPrefixes": [
            "10.0.0.0/16"
        ]
    }
]

```

Figure 18.58: Global VNet peering

rebeladmlive.onmicrosoft.com | Replica sets (preview)
Azure AD Domain Services

Search (Ctrl+I)

+ Add Refresh

Region	Virtual network/Subnet	IP Addresses
East US	REBELVN1/aadds-subnet	10.0.0.5,10.0.0.4

Overview

Activity log

Access control (IAM)

Properties

Secure LDAP

Synchronization

Replica sets (preview) **(selected)**

Health

Notification settings

SKU

Diagnostic settings

Logs

Workbooks

Troubleshoot

New support request

Figure 18.59: Add replica set

Home > rebeladmlive.onmicrosoft.com >

Add a replica set

Save Cancel

Resource group * ①
REBELRG1
Create new

Region * ②
(US) West US

Virtual network * ③
REBELDRVN1
Create new

Subnet * ④
drvmsubnet (10.1.3.0/24)
Manage

Figure 18.60: Configure a replica set

The screenshot shows the 'Replica sets (preview)' section of the Azure AD Domain Services settings. It lists two regions: 'East US' and 'West US'. The 'East US' row shows 'REBELVN1/aadds-subnet' under 'Virtual network/Subnet', '10.0.0.5,10.0.0.4' under 'IP Addresses', and 'Running' under 'Status'. The 'West US' row shows 'REBELDRVN1/drvmsubnet' under 'Virtual network/Subnet', and 'Provisioning' under 'Status'. A red box highlights the 'West US' row.

Region	Virtual network/Subnet	IP Addresses	Status
East US	REBELVN1/aadds-subnet	10.0.0.5,10.0.0.4	Running
West US	REBELDRVN1/drvmsubnet		Provisioning

Figure 18.61: Replica set provisioning

The screenshot shows the 'Deployment' overview for 'AAD_DomainServices_ReplicaSet_20200830224736Z'. It displays a green checkmark indicating 'Your deployment is complete'. Deployment details include: Deployment name: AAD_DomainServices_ReplicaSet_20200830224736Z, Subscription: REBELSS, Resource group: REBELRG1. The start time was 8/30/2020, 11:47:40 PM, and the Correlation ID is b40eac31-f510-452f-ab0f-24ccb7dc716e. There are sections for 'Deployment details' (with a download link) and 'Next steps'.

Figure 18.62: Status of the provisioning task

The screenshot shows the Azure AD Domain Services Replica sets (preview) interface. On the left, there's a navigation sidebar with sections like Overview, Activity log, Access control (IAM), Settings (Properties, Secure LDAP, Synchronization, Replica sets (preview)), Monitoring (Diagnostic settings, Logs, Workbooks), and Support + troubleshooting (Troubleshoot, New support request). The main area displays a table of replica sets:

Region	Virtual network/Subnet	IP Addresses	Status
East US	REBELVN1/aadds-subnet	10.0.0.5,10.0.0.4	Running
West US	REBELDRVN1/drvmsubnet	10.1.3.4	Running

Figure 18.63: Status of the replica set

Code blocks

Command 18.1

```

New-Item
'HKLM:\SOFTWARE\WOW6432Node\Microsoft\.NETFramework\v4.0.30319' -Force
| Out-Null

New-ItemProperty -path
'HKLM:\SOFTWARE\WOW6432Node\Microsoft\.NETFramework\v4.0.30319' -name
'SystemDefaultTlsVersions' -value '1'

.PropertyType 'DWord' -Force | Out-Null

New-ItemProperty -path
'HKLM:\SOFTWARE\WOW6432Node\Microsoft\.NETFramework\v4.0.30319' -name
'SchUseStrongCrypto' -value '1' -PropertyType 'DWord' -Force | Out-
Null

```

```
New-Item 'HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319' -Force | Out-Null

New-ItemProperty -path
'HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319' -name
'SystemDefaultTlsVersions' -value '1' -PropertyType 'DWord' -Force | Out-Null

New-ItemProperty -path
'HKLM:\SOFTWARE\Microsoft\.NETFramework\v4.0.30319' -name
'SchUseStrongCrypto' -value '1' -PropertyType 'DWord' -Force | Out-Null

New-Item
'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server' -Force | Out-Null

New-ItemProperty -path
'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server' -name 'Enabled' -value '1' -PropertyType
'DWord' -Force | Out-Null

New-ItemProperty -path
'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server' -name 'DisabledByDefault' -value 0 -
.PropertyType 'DWord' -Force | Out-Null

New-Item
'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Client' -Force | Out-Null

New-ItemProperty -path
'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Client' -name 'Enabled' -value '1' -PropertyType
'DWord' -Force | Out-Null

New-ItemProperty -path
'HKLM:\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Client' -name 'DisabledByDefault' -value 0 -
.PropertyType 'DWord' -Force | Out-Null
```

```
Write-Host 'TLS 1.2 has been enabled.'
```

Powershell script 18.2

```
$adConnector = "<CASE SENSITIVE AD CONNECTOR NAME>"  
$azureadConnector = "<CASE SENSITIVE AZURE AD CONNECTOR NAME>"  
Import-Module adsync  
$c = Get-ADSyncConnector -Name $adConnector  
$p = New-Object  
Microsoft.IdentityManagement.PowerShell.ObjectModel.ConfigurationParameter "Microsoft.Synchronize.ForceFullPasswordSync", String,  
ConnectorGlobal, $null, $null, $null  
$p.Value = 1  
$c.GlobalParameters.Remove($p.Name)  
$c.GlobalParameters.Add($p)  
$c = Add-ADSyncConnector -Connector $c  
Set-ADSyncAADPasswordSyncConfiguration -SourceConnector $adConnector -  
TargetConnector $azureadConnector -Enable $false  
Set-ADSyncAADPasswordSyncConfiguration -SourceConnector $adConnector -  
TargetConnector $azureadConnector -Enable $true
```

Command 18.3

```
Login-AzAccount  
$DomainServicesResource = Get-AzResource -ResourceType  
"Microsoft.AAD/DomainServices"  
$securitySettings =  
@{"DomainSecuritySettings"=@{ "NtlmV1"="Disabled"; "SyncNtlmPasswords"="Enabled"; "TlsV1"="Disabled" }}  
Set-AzResource -Id $DomainServicesResource.ResourceId -Properties  
$securitySettings -Verbose -Force
```

Command 18.4

```
$vnet1 = Get-AzVirtualNetwork -Name REBELVN1 -ResourceGroupName  
REBELRG1  
$vnet2 = Get-AzVirtualNetwork -Name REBELDRVN1 -ResourceGroupName  
REBELDRRG1  
Add-AzVirtualNetworkPeering -Name REBELVN1toEBELDRVN1 -VirtualNetwork  
$vnet1 -RemoteVirtualNetworkId $vnet2.Id
```

Command 18.5

```
$vnet1 = Get-AzVirtualNetwork -Name REBELVN1 -ResourceGroupName  
REBELRG1  
$vnet2 = Get-AzVirtualNetwork -Name REBELDRVN1 -ResourceGroupName  
REBELDRRG1  
Add-AzVirtualNetworkPeering -Name REBELDRVN1toREBELVN1 -VirtualNetwork  
$vnet2 -RemoteVirtualNetworkId $vnet1.Id
```

Command 18.6

```
$domainname="rebeladmlive.onmicrosoft.com"  
$certlife=Get-Date  
New-SelfSignedCertificate -Subject *.{$domainname} -NotAfter  
$certlife.AddDays(365) -KeyUsage DigitalSignature, KeyEncipherment -  
Type SSLServerAuthentication -DnsName *.{$domainname}, {$domainname}
```

In the above, replace `rebeladmlive.onmicrosoft.com` with your Azure AD DS instance name.

Chapter 19

Figures

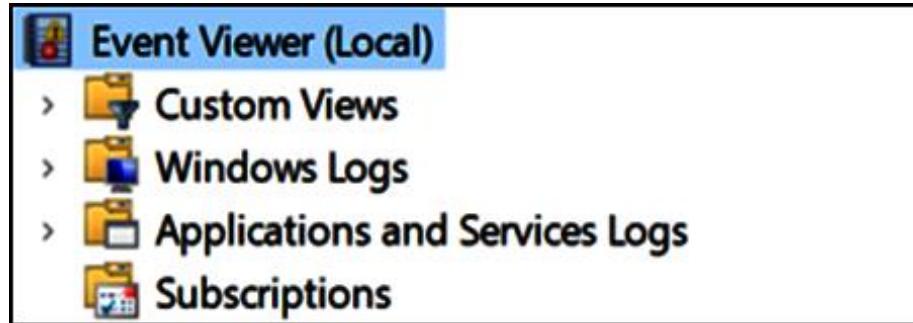


Figure 19.1: Windows Event Viewer

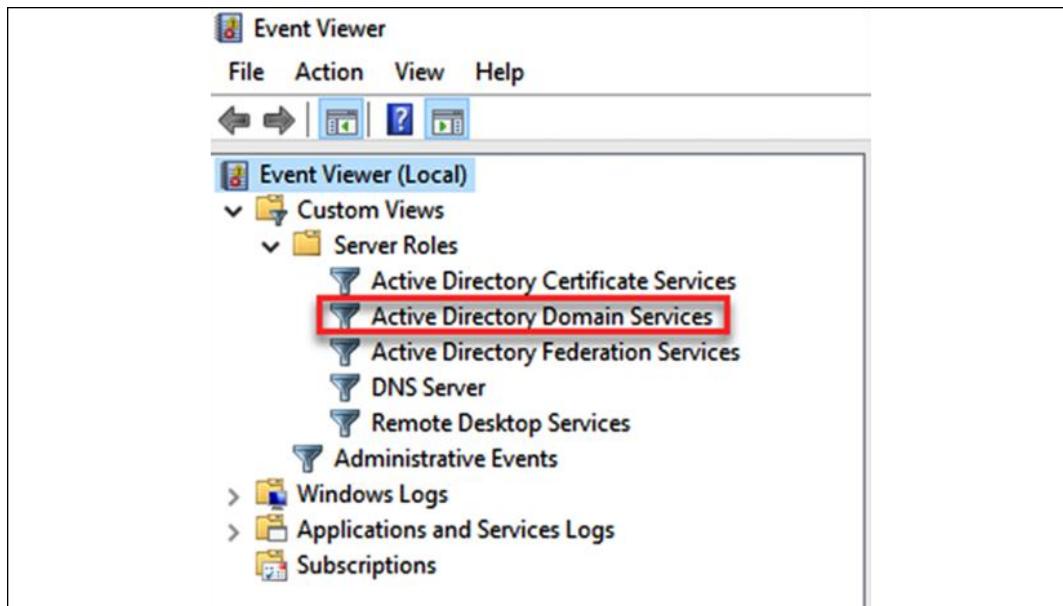


Figure 19.2: Windows Event Viewer Role logs

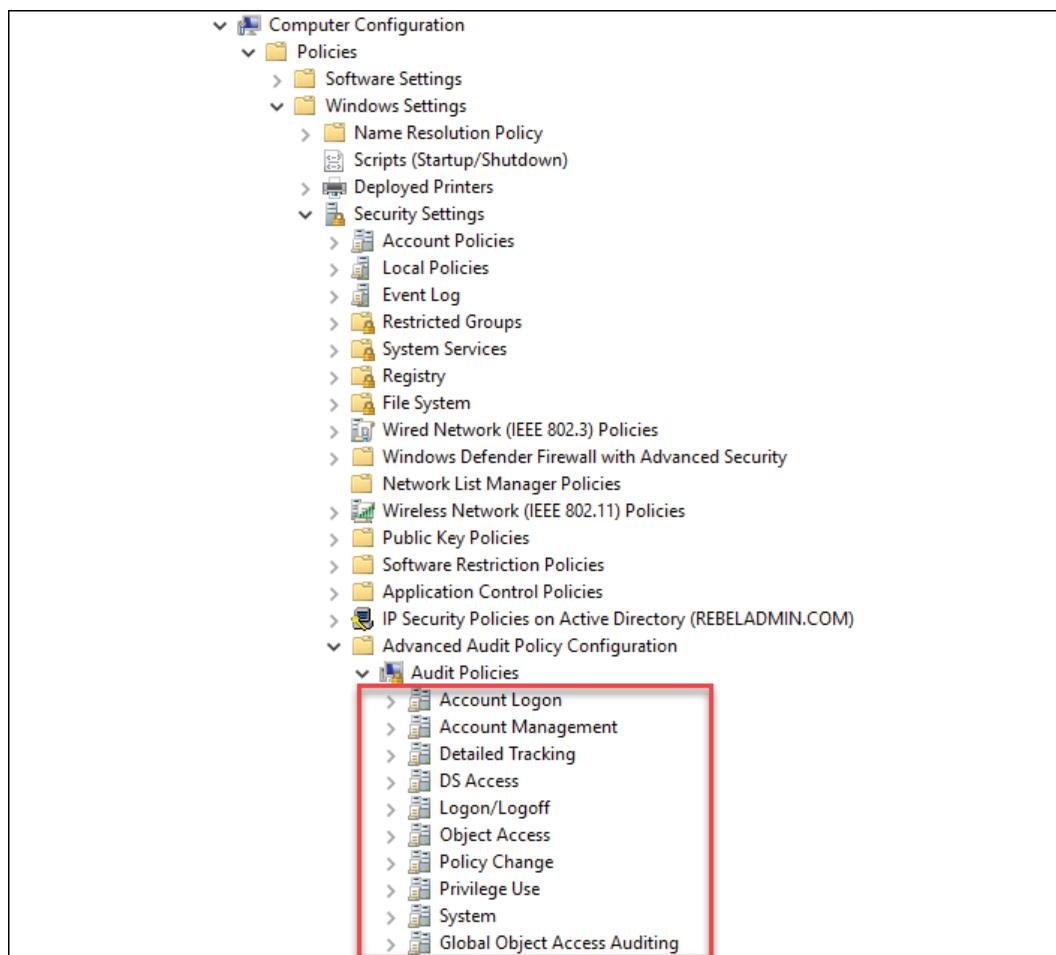


Figure 19.3: Advanced audit logs

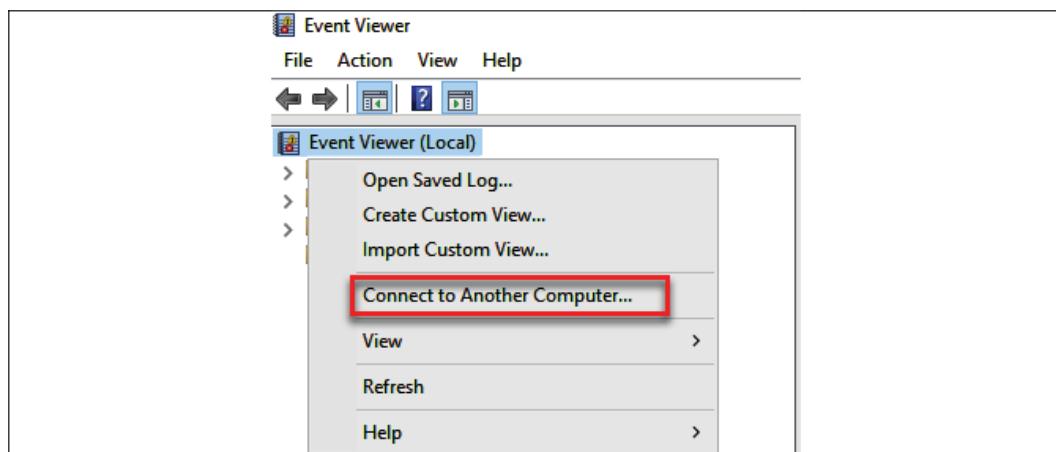


Figure 19.4: Review events on another computer

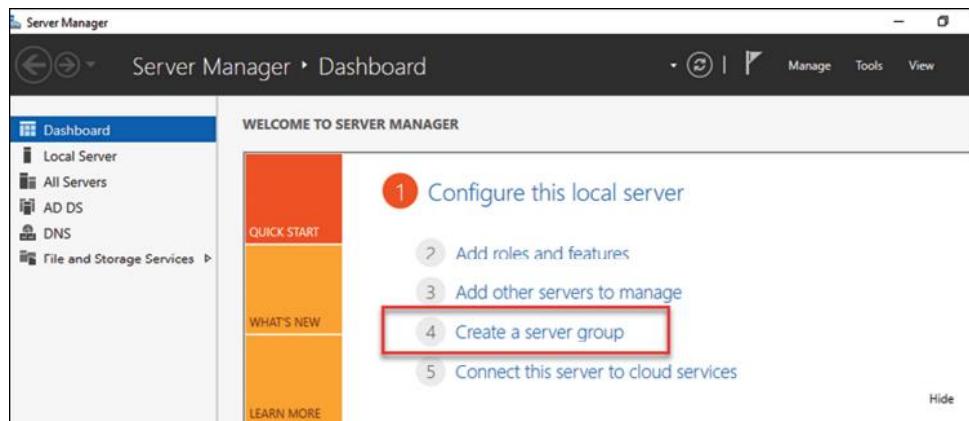


Figure 19.5: Create a server group

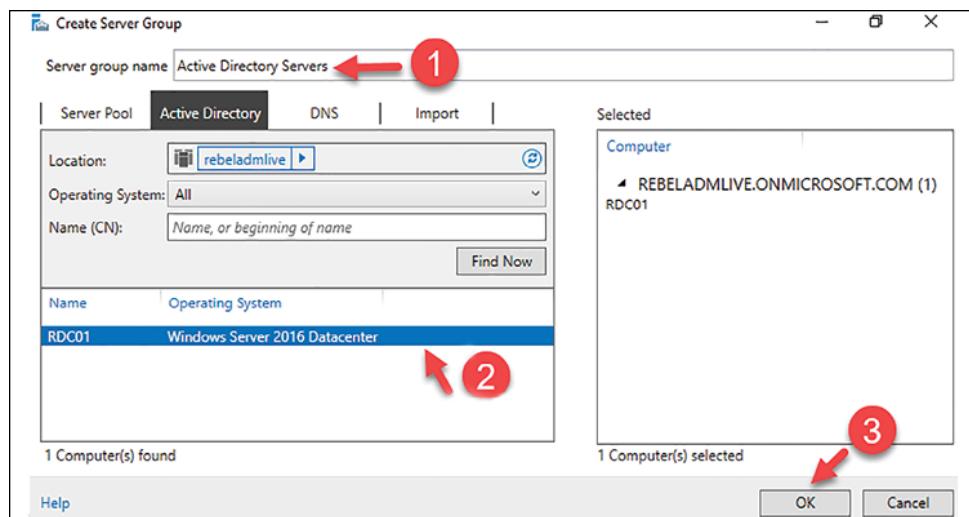


Figure 19.6: Add servers

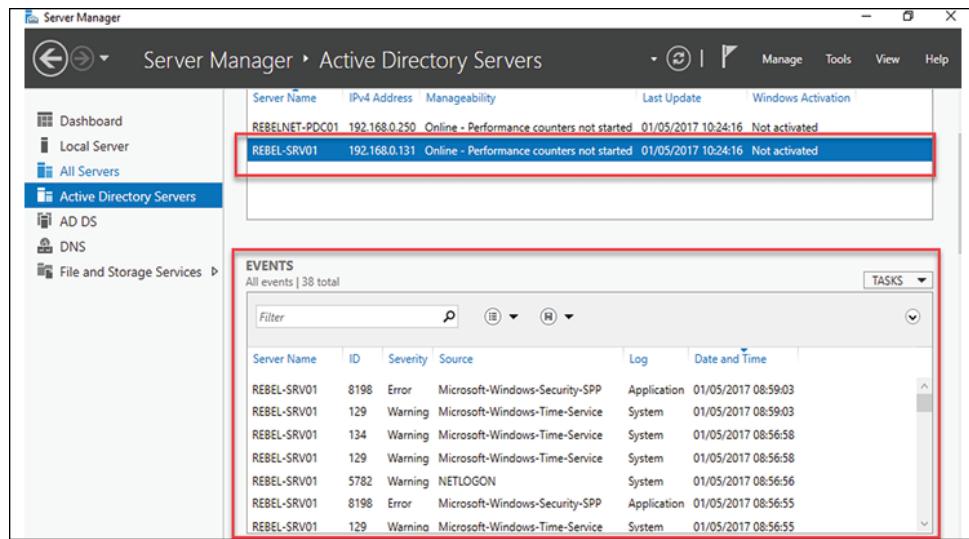


Figure 19.7: Events from the remote server

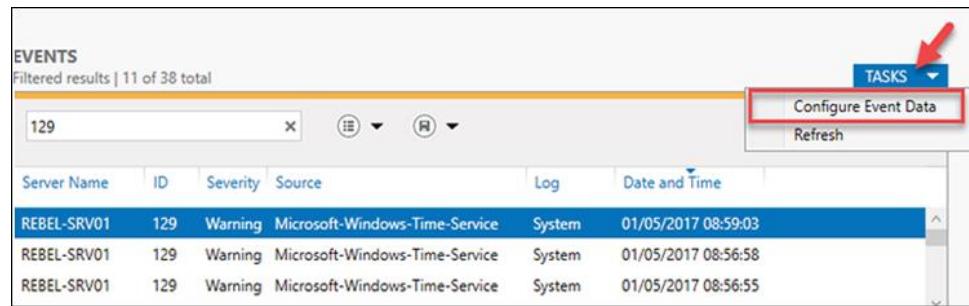


Figure 19.8: Configure Event Data option

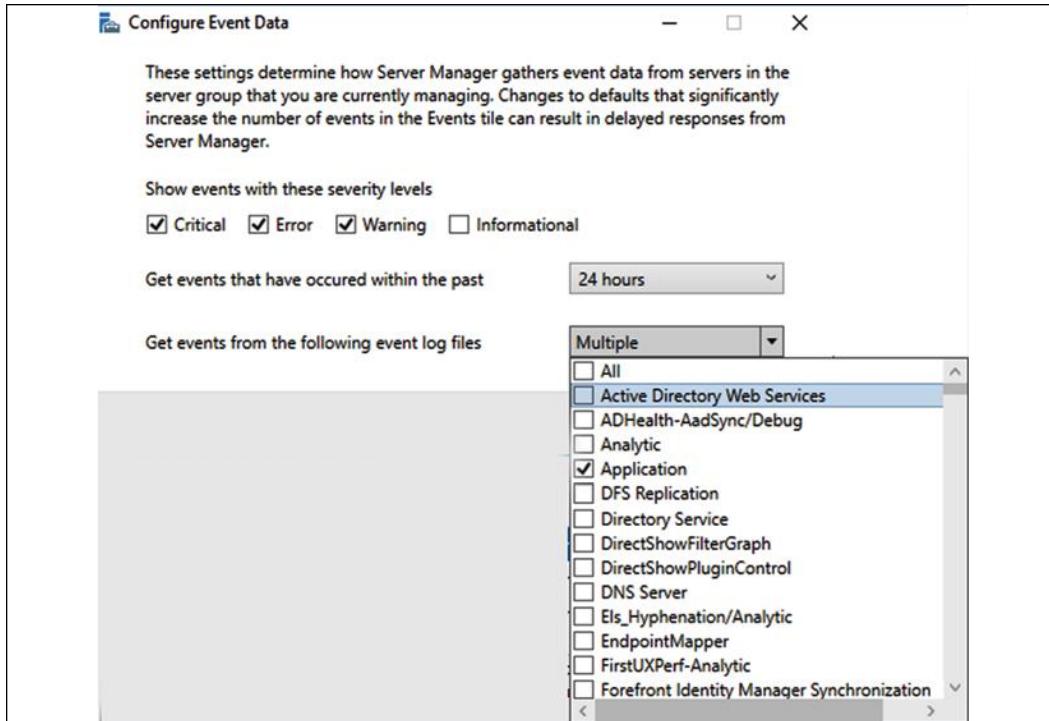


Figure 19.9: Configure Event Data window

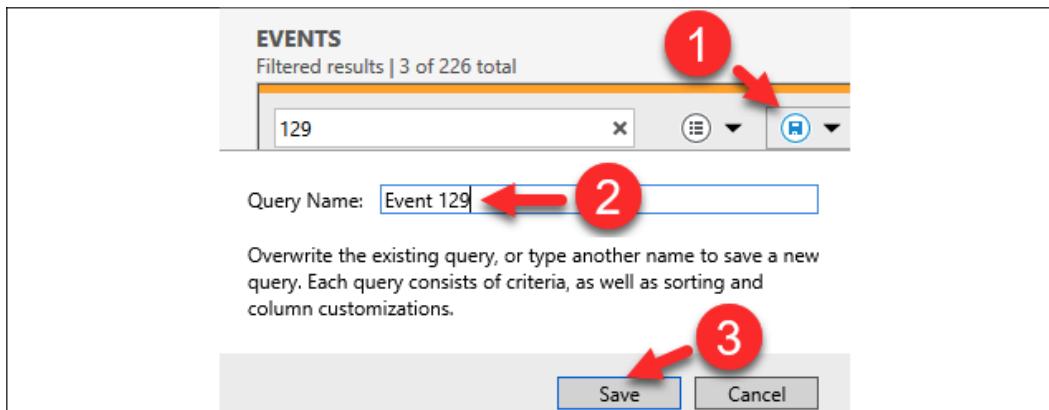


Figure 19.10: Save a query

EVENTS						TASKS
Event 129 32 total						
<input type="text" value="Filter"/> <input type="button" value="P"/> <input type="button" value="E"/> <input type="button" value="R"/>						
Server Name	ID	Severity	Source	Event 129	Log	Date and Time
REBEL-SRV01	129	Warning	Microso		System	01/05/2017 08:59:03
REBEL-SRV01	134	Warning	Microso		System	01/05/2017 08:56:58
REBEL-SRV01	129	Warning	Microso		System	01/05/2017 08:56:58

Figure 19.11: Access a saved query

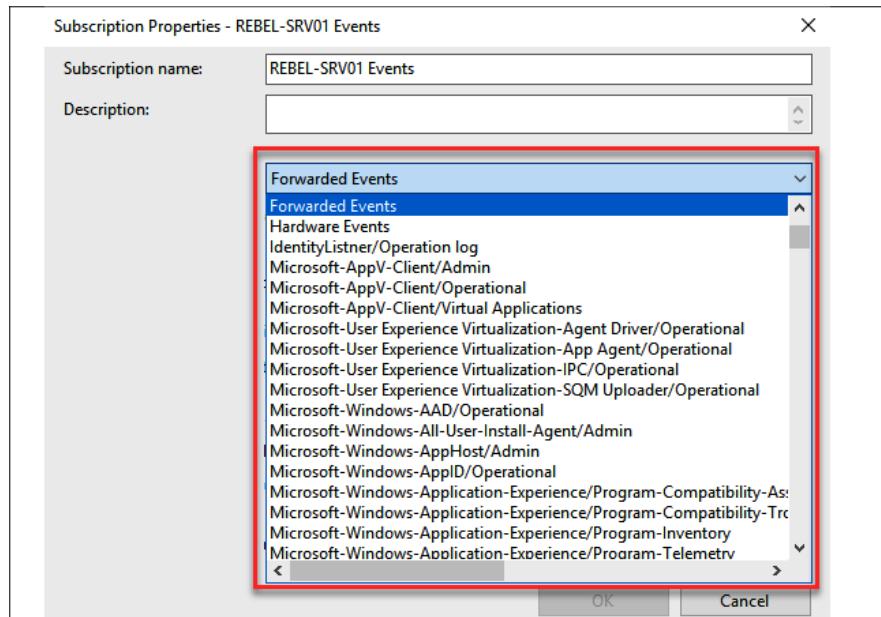


Figure 19.12: Event subscription

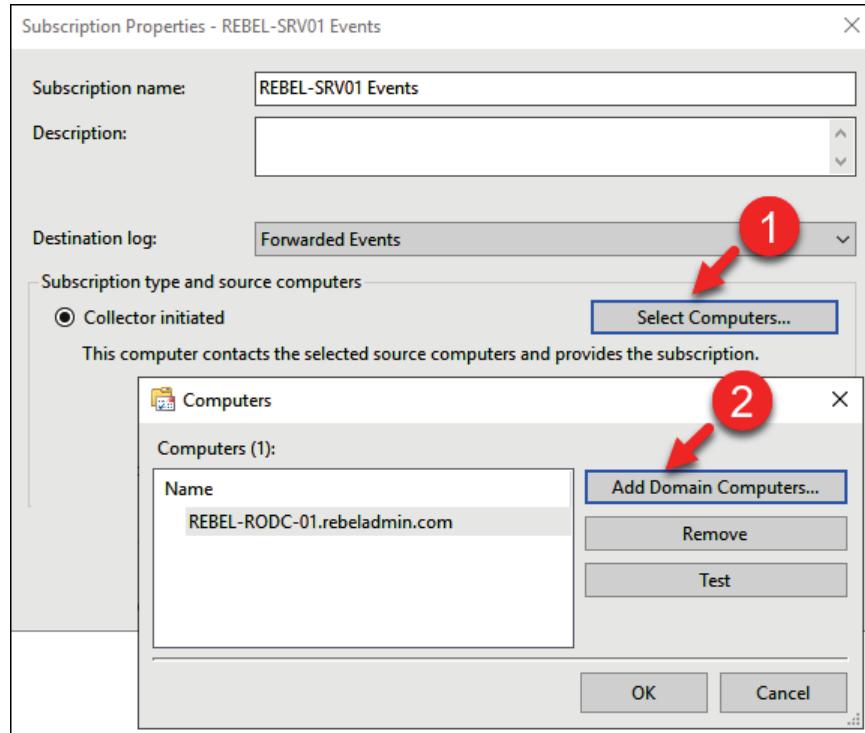


Figure 19.13: Select source computer

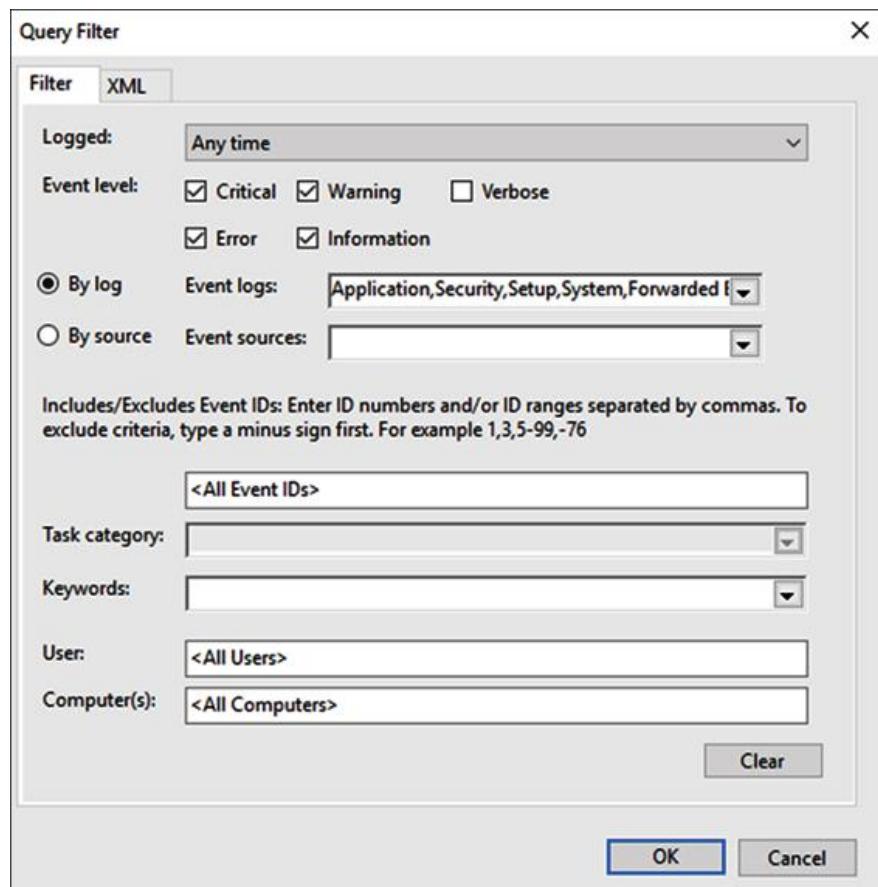


Figure 19.14: Query events

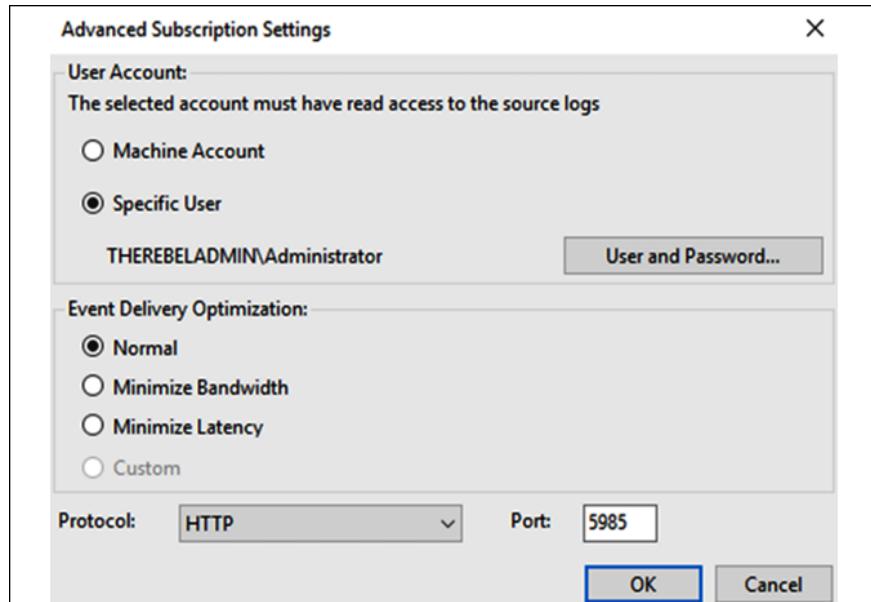


Figure 19.15: Advanced Subscription Settings

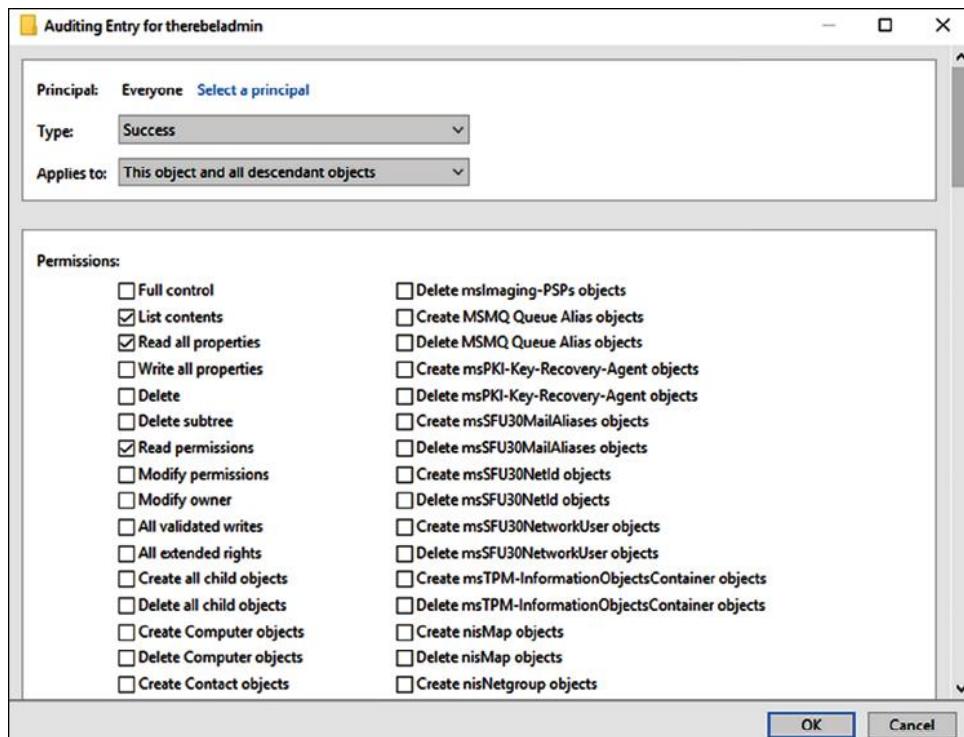


Figure 19.16: SACL entry

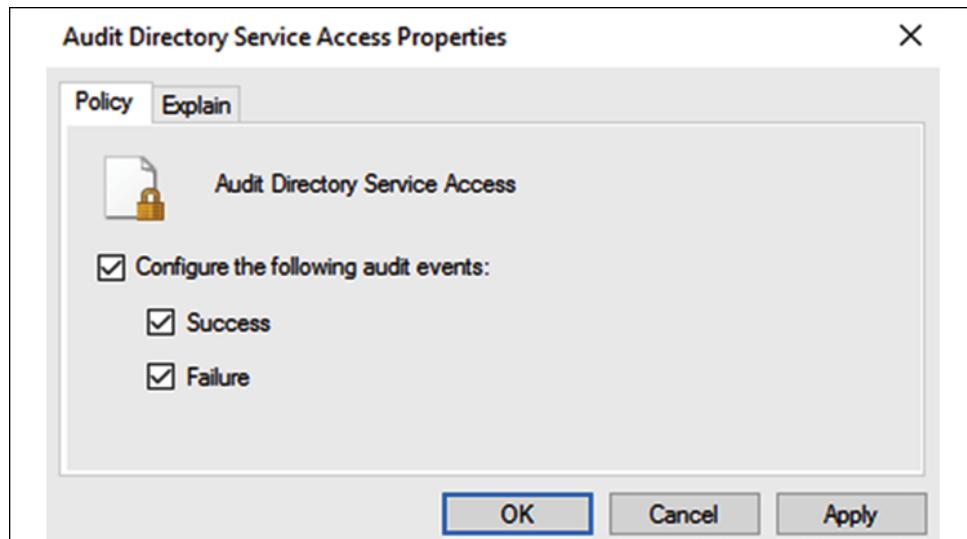


Figure 19.17: Audit Directory Service Access events

Subcategory	Audit Events
101 Audit Detailed Directory Service Replication	Success and Failure
102 Audit Directory Service Access	Success and Failure
103 Audit Directory Service Changes	Success and Failure
104 Audit Directory Service Replication	Success and Failure

Figure 19.18: Audit categories

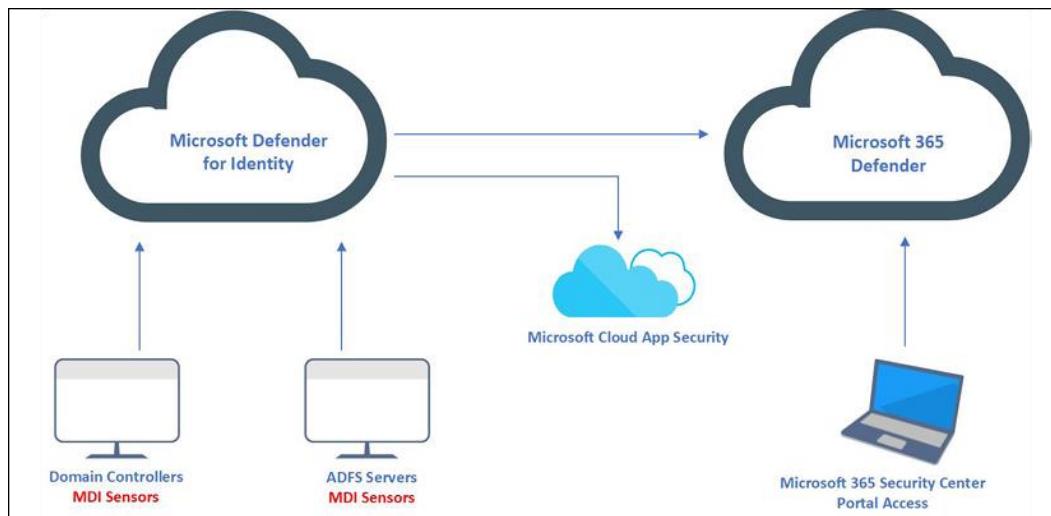


Figure 19.19: MDI architecture

Application Management	Processes in...	Manual
AppX Deployment Service (AppXSVC)	Provides inf...	Manual
Auto Time Zone Updater	Automatica...	Disabled
Azure AD Connect Health Sync Insights Service	Azure AD C...	Running Automatic (D...
Azure AD Connect Health Sync Monitoring Service	Azure AD C...	Running Automatic (D...
Background Intelligent Transfer Service	Transfers fil...	Manual
Background Tasks Infrastructure Service	Windows in...	Running Automatic

Figure 19.20: Azure AD Connect health services

The screenshot shows the 'Contoso - Azure AD Connect' dashboard under 'Azure Active Directory'. On the left, there's a navigation menu with items like 'Users', 'Groups', 'Organizational relationships', etc., and 'Azure AD Connect' is selected. The main area is titled 'SYNC STATUS' and shows 'Sync Status' as Enabled, 'Last Sync' as less than 1 hour ago, and 'Password Hash Sync' as Enabled. Below that is 'USER SIGN-IN' with 'Federation' as Disabled, 'Seamless single sign-on' as Disabled, and 'Pass-through authentication' as Disabled. There are sections for 'ON-PREMISES APPLICATIONS' and 'HEALTH AND ANALYTICS'. A red arrow points to the 'Monitor your on-premises identity infrastructure and synchronization services in the cloud. Azure AD Connect Health' link.

Figure 19.21: Azure AD Connect Health access

The screenshot shows the 'Azure Active Directory Connect Health - Quick start' page. The left sidebar has links for 'Quick start', 'Sync errors', 'Sync services', 'AD FS services', 'AD DS services', 'Configure', 'Settings', 'Role based access control (IAM)', 'Troubleshoot', and 'New support request'. The main content area has a 'Get a free Azure AD Premium trial...' banner. It features a 'What's New' section about Azure AD Connect Health for Sync. Below that is a 'Get tools' section with a link to 'Download Azure AD Connect Health Agent for AD DS', which is highlighted with a red box. Other sections include 'Provide feedback' and 'Learn more'.

Figure 19.22: Download Azure AD Connect Health Agent for AD DS

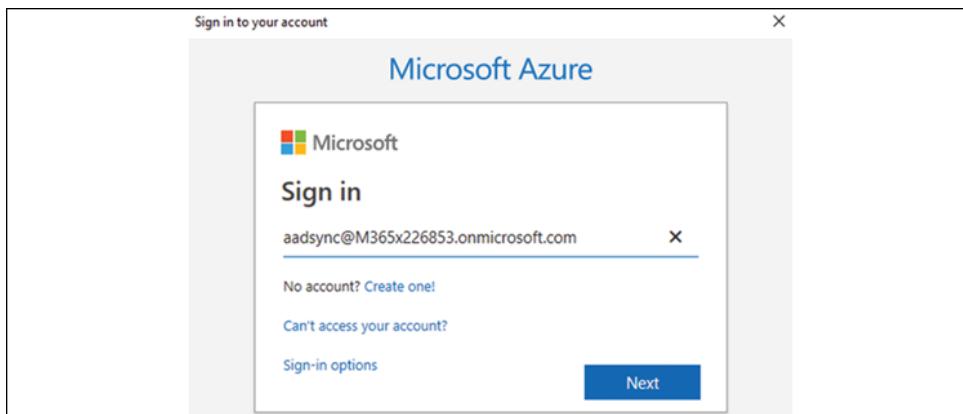


Figure 19.23: Azure AD login

```
Administrator: Windows PowerShell
2019-06-09 17:46:38.941 AHealthServiceUri (ARM): https://management.azure.com/providers/Microsoft.DHybridHealthService/
2019-06-09 17:46:38.941 AdHybridHealthServiceUri: https://adds.aadconnecthealth.azure.com/
2019-06-09 17:46:38.957 AHealthServiceUri (ARM): https://management.azure.com/providers/Microsoft.DHybridHealthService/
2019-06-09 17:46:38.957 AdHybridHealthServiceUri: https://adds.aadconnecthealth.azure.com/
2019-06-09 17:46:39.535 AHealthServiceApiVersion: 2014-01-01
2019-06-09 17:49:09.957 Detecting AdDomainService roles...
2019-06-09 17:49:10.114 Detected the following role(s) for M365x226853.onmicrosoft.com:
2019-06-09 17:49:10.114          Active Directory Domain Services
2019-06-09 17:49:19.379 Acquiring Monitoring Service certificate using tenant.cert
2019-06-09 17:49:23.019 Successfully acquired and stored Monitoring Service certificate: Subject=CN=PDC01, CN=F609ba2e-559f-435f-ba98-1d421b9d25cd, OU=Microsoft ADFS Agent, Issuer=CN=Microsoft PolicyKeyService Certificate Authority, Thumbprint=13C6FDCC094F5890D7936E61F1BA41A4E00FB89A
2019-06-09 17:49:23.035 Fetched and stored agent credentials successfully...
2019-06-09 17:49:23.035 Starting agent services...
2019-06-09 17:49:38.973 Started agent services successfully...
2019-06-09 17:49:49.426 Agent registration completed successfully.

Detailed log file created in temporary directory:
C:\Users\dfrancis\AppData\Local\Temp\AdHealthAddsAgentConfiguration.2019-06-09_18-46-38.log
PS C:\Windows\system32>
```

Figure 19.24: Azure AD Connect Health service registration

SERVICE NAME	ACTIVE ALERTS	LAST UPDATED	STATUS
M365x226853.onmicrosoft.com	0	09/06/2019, 21:27:52	Healthy

Figure 19.25: Sync errors and Sync services

Overview

Azure Active Directory Connect Servers

1	1
PDC01	Healthy

Operations

Alerts

0 active
Active 0
Resolved from last 24 hours 0

Last export to Azure AD

Exported 6/9/2019, 9:16:08 PM

Sync Error

AadSyncService-M365x226853....

Figure 19.26: Overview of Azure AD Connect Health

The screenshot shows the 'Azure Active Directory Connect Health - Sync errors' page for the 'Contoso' tenant. The left sidebar lists navigation options: 'Quick start', 'Azure Active Directory Connect (Sync)', 'Sync errors' (which is selected and highlighted in blue), 'Sync services', 'Active Directory Federation Services', 'AD FS services', 'Active Directory Domain Services', and 'AD DS services'. The main content area is titled 'Sync Error by Type' and displays two categories: 'Duplicate Attribute' and 'Data Mismatch'. Both categories show a count of '0'.

Figure 19.27: Sync errors

The screenshot shows the 'Azure Active Directory Connect Health - AD DS services' page for the 'Contoso' tenant. The left sidebar lists navigation options: 'Quick start', 'Azure Active Directory Connect (Sync)', 'Sync errors', 'Sync services', 'Active Directory Federation Services', 'AD FS services', 'Active Directory Domain Services' (which is selected and highlighted in red), and 'AD DS services'. The main content area displays a table with one row, showing details for the service 'M365x226853.onmicrosoft.com'. The table columns are 'SERVICE NAME', 'ACTIVE ALERTS', 'LAST UPDATED', and 'STATUS'. The status is listed as 'Healthy'.

Find...	SERVICE NAME	ACTIVE ALERTS	LAST UPDATED	STATUS
	M365x226853.onmicrosoft.com	0	09/06/2019, 21:37:43	Healthy

Figure 19.28: AD DS services

The screenshot shows the Azure Active Directory Connect Health - AD DS services interface for the domain **M365x226853.onmicrosoft.com**. The top navigation bar includes links to Home, Contoso - Azure AD Connect, and Azure Active Directory Connect Health - AD DS services.

Monitoring:

- Domains:** 1 DOMAIN
- Sites:** 1 SITES

Replication Status: M365x226853.onmicrosoft.com

Operations:

- Alerts:** M365x226853.onmicrosoft.com
 - 0 active**
- Active:** 0
- Resolved from last 24 hours:** 0

Monitoring:

DCs with errors: 0

Figure 19.29: health

Code blocks

Command 19.1

```
Add-ADGroupMember -identity 'Event Log Readers'  
-members REBELNET-PDC01$
```

Code 19.1

```
wEvtutil sl security  
/ca:'O:BAG:SYD:(A;;0xf0005;;;SY)(A;;0x5;;;BA)(A;;0x1;;;S-1-5-32-  
573)(A;;0x1;;;S-1-5-20)'
```

O:BAG:SYD:(A;;0xf0005;;;SY)(A;;0x5;;;BA)(A;;0x1;;;S-1-5-32-573)(A;;0x1;;;S-1-5-20) contains READ permission settings for network service account (A;;0x1; ;). In the preceding code, the SID value for the network service account is (S-1-5-20), and

the channel access value is

```
(O:BAG:SYD:(A;;0xf0005;;;SY)(A;;0x5;;;BA)(A;;0x1;;;S-1-5-32-573)).
```

Command 19.2

```
Get-EventLog -List
```

Command 19.3

```
Get-EventLog -LogName 'Directory Service' | fl
```

Command 19.4

```
Get-EventLog -Newest 5 -LogName 'Directory Service' -EntryType Error
```

Command 19.5

```
Get-EventLog -Newest 5 -LogName 'Directory Service' -EntryType Error -  
After (Get-Date).AddDays(-1)
```

Command 19.6

```
Get-EventLog -Newest 5 -LogName 'Directory Service' -ComputerName  
'REBEL-SRV01' | fl -Property *
```

Command 19.7

```
Get-EventLog -Newest 5 -LogName 'Directory Service' -ComputerName  
"localhost", "REBEL-SRV01"
```

Command 19.8

```
Get-EventLog -LogName 'Directory Service' -Source "NTDS_KCC"
```

Command 19.9

```
Get-EventLog -LogName 'Directory Service' | where {$_.eventID -eq  
1000}
```

Tables

Event ID	Event message
4662	An operation was performed on an object

Table 19.1

Event ID	Event message
5136	A directory service object was modified.
5137	A directory service object was created.
5138	A directory service object was undeleted.
5139	A directory service object was moved.
5141	A directory service object was deleted.

Table 19.2

Event ID	Event message
4932	Synchronization of a replica of an AD naming context has begun.
4933	Synchronization of a replica of an AD naming context has ended.

Table 19.3

Event ID	Event message
4928	An AD replica source naming context was established.
4929	An AD replica source naming context was removed.
4930	An AD replica source naming context was modified.
4931	An AD replica destination naming context was modified.
4934	Attributes of an AD object were replicated.
4935	Replication failure start.
4936	Replication failure end.
4937	A lingering object was removed from a replica.

Table 19.4

Protocol	TCP/UDP	Port	To/From	Direction
SSL	TCP	443	Defender for Identity cloud services	Outbound
SSL	TCP	444	Sensor service	Both
DNS	TCP and UDP	53	Sensors to DNS Servers	Outbound
Netlogon	TCP/UDP	445	Sensors to all devices	Outbound
RADIUS	UDP	1813	RADIUS to sensors	Inbound

NTLM over RPC	TCP	135	Sensors to all devices	Outbound
NetBIOS	UDP	137	Sensors to all devices	Outbound
TLS to RDP	TCP	3389	Sensors to all devices	Outbound

Table 19.5