Mastering Identity and Access Management with Microsoft Azure

Second Edition

Empower users by managing and protecting identities and data

Packt>

www.packt.com

Jochen Nickel

Mastering Identity and Access Management with Microsoft Azure
Second Edition

Empower users by managing and protecting identities and data

Jochen Nickel

Packt

BIRMINGHAM - MUMBAI

Mastering Identity and Access Management with Microsoft Azure
Second Edition

Copyright © 2019 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Gebin George
Acquisition Editor: Rahul Nair
Content Development Editor: Deepti Thore
Technical Editor: Mamta Yadav
Copy Editor: Safis Editing
Project Coordinator: Nusaiba Ansari
Proofreader: Safis Editing
Indexer: Tejal Daruwale Soni
Graphics: Jisha Chirayil
Production Coordinator: Aparna Bhagat

First published: September 2016
Second edition: February 2019

Production reference: 1250219

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-78913-230-4

www.packtpub.com

mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

Packt.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Contributors

About the author

Jochen Nickel is a Cloud, Identity and Access Management Solution Architect with a clear focus and in-depth technical knowledge of Identity and Access Management. He is currently working for inovit GmbH in Switzerland leading and executing projects in the field of Identity and Access Management including Data Classification and Information protection. Jochen is focused on Microsoft Technologies, especially in the Enterprise Mobility + Security Suite, Office 365 and Azure. He is an established speaker at many technology conferences like Azure Bootcamps, TrustInTech Meetups or the Experts Live Switzerland and Europe.

About the reviewer

Kasam Shaikh, a Microsoft Azure enthusiast, is a seasoned professional with a "can do" attitude and 11 years of industry experience, working as a cloud architect with one of the leading IT companies in Mumbai, India. He is a certified Azure architect, YouTuber, recognized as an MVP by a leading online community, as well as a global AI speaker, and has authored books on Azure Cognitive, Azure Bots, and Microsoft Bot frameworks. He is the founder of Dear Azure, (AZ-INDIA) community, the fastest-growing online community for learning Azure.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit `authors.packtpub.com` and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Preface

Mastering Identity and Access Management with Microsoft Azure is a crisp and practical, hands-on guide containing project scenarios/illustrations that is tailored to genuine hybrid and cloud-only challenges. Developers, security specialists, IT consultants, and architects are the audience for this book. With it, you get a complete companion for solving key topics in the field of identity and access management through all related Microsoft technologies and practice-related crisp and clear content that helps you to put the theory into practice. The book delves into the Microsoft 365 Security and Compliance plans and other Azure services related to identity and access management topics.

The book is divided into three parts. In the first part, crucial identity management topics are covered, such as identity synchronization as a whole, including monitoring and protection topics, in a cloud-only and hybrid world. The second part provides all the essentials and in-depth knowledge pertaining to the different authentication methods you can use and how you can securely publish and expose your applications with on-premise technologies and the Azure AD feature set. The final part of the book focuses entirely on the Microsoft information protection technologies. Another highlight is the more than 40 playbooks you receive to support the learning process through practical tips. With this great resource, you get an information package that also covers the functionality of Windows 10 and Windows Server 2016/2019.

How does this edition differ from the first edition of the book, and why a second edition with more than 85% new content?

First of all, many thanks to all the readers and the valuable feedback I received. I was happy to listen!

Since writing the first edition of the book back in 2016, many features have been completely updated, added, changed, or even removed. The Microsoft Azure world is changing very rapidly, from a pure infrastructure to an object and service-oriented environment. For this reason, it is necessary to include a variety of developmental aspects in the book. Some functions are currently changing their entitlement entirely to the cloud.

However, no overall solution for sustainable identity and access management in a hybrid cloud environment is currently available to fulfill all the different aspects. For this reason, the basics for individual services must be developed to ensure a better shift of the functions.

Another important reason for me to write an updated edition was that I heard from readers and workshop attendees that they require more technical guidance and less information on the decision manager side. This brought me to an approach whereby I provide more than 40 hands-on guides in the book, where you can test all the related information in a practical and guided manner. Furthermore, our workshop attendees and customers found it very hard to find qualified and working lab examples in a compressed form to save time and effort.

Many of you and our attendees loved the structure of the three scenarios in the first book. Frequently, however, I received a request to provide the theory and practical guidance in technology or topic-based flows so as to make it easier to follow, if you are just interested in specific topics, or if you want to use the book as a living reference.

At the time of writing the first book, the Azure information protection technology was not available in the complete approach that it is available today. Since this technology is now mature and an integral aspect of access management, in my view, additional chapters for this topic are an absolute necessity.

Windows Server 2019 is also available to use, so I updated the book to work with the new server version, with a primary focus on hybrid cloud scenarios.

Who this book is for

This book is designed for cyber security specialists, system and security engineers, developers and IT consultants/architects who wish to plan, design, and implement identity and access management solutions with the help of Microsoft Azure technology.

What this book covers

`Chapter 1`, *Building and Managing Azure Active Directory*, explains how to configure a suitable Azure AD tenant for a cloud-only approach. You will also learn how to configure and manage users, groups, roles, and administrative units to provide a user and group-based application and self-service access, including the related audit functionality.

`Chapter 2`, *Understanding Identity Synchronization*, explains the most important identity synchronization scenarios and tools for successful implementation of a complete hybrid identity life cycle management. We will run through the different processes, the Active Directory user account cleanup for a hybrid environment, and all the crucial identity synchronization aspects and steps in Azure Active Directory Connect.

`Chapter 3`, *Exploring Advanced Synchronization Concepts*, teaches you the advanced synchronization concepts. In particular, we will look into the synchronization rules and the declarative provisioning and expressions concept and use them directly in real-world examples.

`Chapter 4`, *Monitoring Your Identity Bridge*, explains the various monitoring capabilities for the identity bridge that's constructed by Azure AD Connect, the Active Directory itself and, if used, the **Active Directory Federations Services** (**ADFS**) and the Web Application Proxy. We'll investigate the Azure AD Monitoring and Logs' functionalities, the Azure AD Health Service, and the Azure Security Center.

`Chapter 5`, *Configuring and Managing Identity Protection*, demonstrates how to protect your identities against today's attacks. We will work through the different cloud services that can help you protect your environment so that you can plan and implement the features for your requirements.

`Chapter 6`, *Managing Authentication Protocols*, teaches you the basic authentication protocols you need to know for handling ADFS and Azure AD integrations. Additionally, you will benefit from a vast array of validated and recommended material to facilitate a deep dive into every critical authentication and authorization protocol.

`Chapter 7`, *Deploying Solutions on Azure AD and ADFS*, explains how to configure Azure AD and ADFS to handle your application requirements. You will install the service and the authentication platform to gather all the knowledge required in order to emerge victorious in this field of technology.

`Chapter 8`, *Using the Azure AD App Proxy and the Web Application Proxy*, covers the publishing of applications through the Azure AD Application Proxy and the Windows Server Web Application Proxy. We will configure a number of applications, including the first conditional access scenarios.

`Chapter 9`, *Deploying Additional Applications on Azure AD*, explains the concept of single- and multi-tenant applications and the differences between the two. Furthermore, you will configure the two types of application, including the transition process from single- to multi-tenant.

`Chapter 10`, *Exploring Azure AD Identity Services*, explains the different Azure AD identity services and ADFS as on-premise identity services. We will look at the Azure AD B2B and B2C functionality and explain the main concepts regarding these technologies.

`Chapter 11`, *Creating Identity Life Cycle Management on Azure*, covers different identity life cycle scenarios. With a strong focus on a complete Azure AD B2B management, we will provide you with all the requisite information and configuration tasks to offer comfortable and secure application access to your users.

`Chapter 12`, *Creating a New Security Culture*, explains why organizations need to build a strong security culture to provide a suitable information protection solution. You will get a clear and crisp overview to understand the three key factors and the four main pillars of a strong security culture.

`Chapter 13`, *Identifying and Detecting Sensitive Data*, teaches you why identifying and detecting sensitive data is a critical process inside an information protection solution. You will work through all the related technologies and configure a number of solutions.

`Chapter 14`, *Understanding Encryption Key Management Strategies*, explains how to use the three crucial, and different, deployment models and the role played by the Azure Key Vault service. Furthermore, you will learn how the Azure Rights Management Services uses the various keys on client applications.

`Chapter 15`, *Configuring Azure Information Protection Solutions*, shows you how to start an Azure information protection project and provides you with best practices and configuration tips for successful implementation.

`Chapter 16`, *Azure Information Protection Development Overview*, provides you with a solid foundation for using the Microsoft Information Protection developer resources for gathering more in-depth knowledge to handle this service in terms of troubleshooting or developing your extension.

To get the most out of this book

To use the book efficiently, you should have some understanding of security solutions, Active Directory, access privileges/rights, and authentication methods. Programming knowledge is not required but will be helpful for using PowerShell or working with APIs to customize your solutions. Working through the first edition of the book is not a requirement for following the chapters in this book.

During the book, we will work entirely on the Azure platform itself. The only requirement is to have an internet connection and a Microsoft or Apple client computer. The labs can be undertaken free of charge for the duration of the several trial versions we use. We highly recommend that you shut down your virtual machines on Azure to save the runtime for working with your practical guidance. In `Chapter 7`, Deploying Solutions on Azure AD and ADFS, we will provide you the architecture overview with all the requisite information for sizing and the different products we use and will reference in the chapters. We also provide you the guidance to create public certificates with Let's Encrypt. One small cost requirement exists. If you want to run all the different labs, you need to have three public DNS domains registered, including access to the related public DNS. Bear in mind that this lab is for studying and testing functionality and not a representation of a productive environment. Follow the instructions in the chapters to arrange the correct resources. All the scripts and demo files are covered in the example code files, which you can download on the web page provided in the download the example code files section.

Download the example code files

You can download the example code files for this book from your account at `www.packt.com`. If you purchased this book elsewhere, you can visit `www.packt.com/support` and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at `www.packt.com`.
2. Select the **SUPPORT** tab.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at `https://github.com/PacktPublishing/Mastering-Identity-and-Access-Management-with-Microsoft-Azure.Second-Edition`. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `https://www.packtpub.com/sites/default/files/downloads/9781789132304_ColorImages.pdf`

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "Open your Visual Studio and the solution file from the code package named `WebApp-RoleClaims-DotNet.sln`."

A block of code is set as follows:

```
$ServicePrincipalName="RMSPowerShell"
Connect-AadrmService
$bposTenantID=(Get-AadrmConfiguration).BPOSId
Disconnect-AadrmService
Connect-MsolService
New-MsolServicePrincipal -DisplayName $ServicePrincipalName
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
$cred = Get-Credential
Install-AIPScanner -SqlServerInstance YD1APP01 -
ServiceUserCredentials $cred
```

Any command-line input or output is written as follows:

```
$ImportFile = Import-csv "$dirpath\ADUsers.csv"
$TotalImports = $importFile.Count
```

Bold: Indicates a new term, an important word, or words that you see on screen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Click on **Add New Rule** and select **Inbound**."

Warnings or important notes appear like this.

Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at `customercare@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packt.com/submit-errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packt.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit `packt.com`.

1
Section 1: Identity Management and Synchronization

In this first section of the book, you will explore and use several Microsoft Identity and Access Management Service offerings. You will look into Identity Synchronization processes in detail as well.

The following chapter will be covered in this section:

- `Chapter 1`, Building and Managing Azure Active Directory
- `Chapter 2`, Understanding Identity Synchronization
- `Chapter 3`, Exploring Advanced Synchronization Concepts
- `Chapter 4`, Monitoring Your Identity Bridge
- `Chapter 5`, Configuring and Managing Identity Protection

1
Building and Managing Azure Active Directory

Working with the several **Software-as-a-Service** (**SaaS**) offerings such as Office 365, Dynamics CRM or Visual Studio Online requires well-managed identities and an excellent basic structure in the Azure **Active Directory** (**AD**) that builds the heart of these solutions. You, as an administrator, need to provide a stable identity and access management platform to manage these services.

This chapter explains how to configure a suitable Azure AD tenant, which we use throughout the whole book to explore, understand, and configure the different features and functions in the field of identity and access management with Microsoft Azure. We start with the cloud-only components, followed in the next chapters by the hybrid identity and access management approach.

In this chapter, we go directly to the configuration and learn how to configure and manage users, groups, roles, and administrative units to provide a user and group-based application and self-service access, including the audit functionality. The chapter focuses on the following :

- Implementation scenario overview
- Implementing a solid Azure Active Directory
- Creating and managing users and groups
- Assigning roles and administrative units
- Protecting your administrative accounts
- Providing user and group-based application access
- Activating password reset

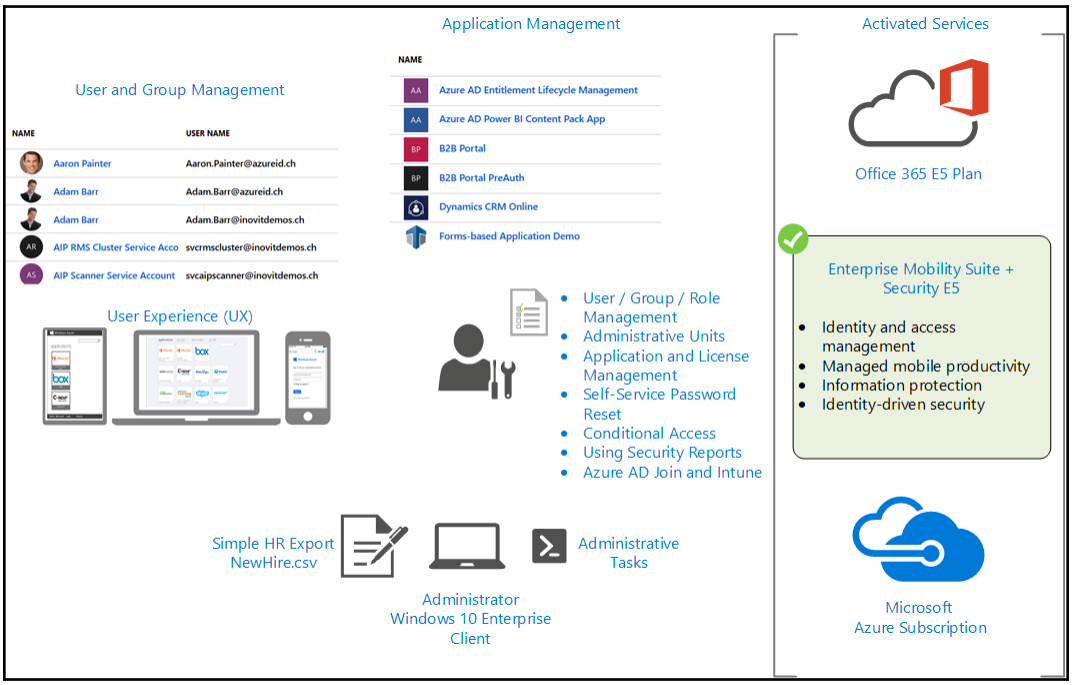
- Using standard security monitoring
- Integrating the Azure AD Join for Windows 10 clients
- Configuring a custom domain
- Configure Azure AD Domain Services

Now, we can introduce the implementation scenario.

Implementation scenario overview

After completing the next configuration tasks, you will see the rich functionality of Microsoft Azure in the field of identity and access management, starting with cloud identities. You can demonstrate the different capabilities in your own Microsoft Azure environment. The guidance will focus on the most essential feature sets to give you an idea about their capabilities. We will start to use the default directory, which we call `domain.onmicrosoft.com` for now, and will change it later to a custom domain name. Domain stands for your desired name like `example.com`, this is also used for the userPrincipalName of the users in this chapter, e.g. don.hall@doamin.onmicrosoft.com is represented in the chapter by my example domain called inovitcloudlabs. Be aware that this name will be visible in different applications, such as SharePoint Online and Skype for Business, to the end user. We recommend the company name without the company form, for instance, inovit GmbH would be `inovit.onmicrosoft.com`. Use a different name for your tests, so that the domain for a productive environment stays free. This configuration will be the base for all further scenarios in the book. For this reason, we use an **Azure**, **Enterprise Mobility Suite**, and **Office 365** subscription to use all the available features.

The following figure shows the different main areas we will focus on in this chapter:



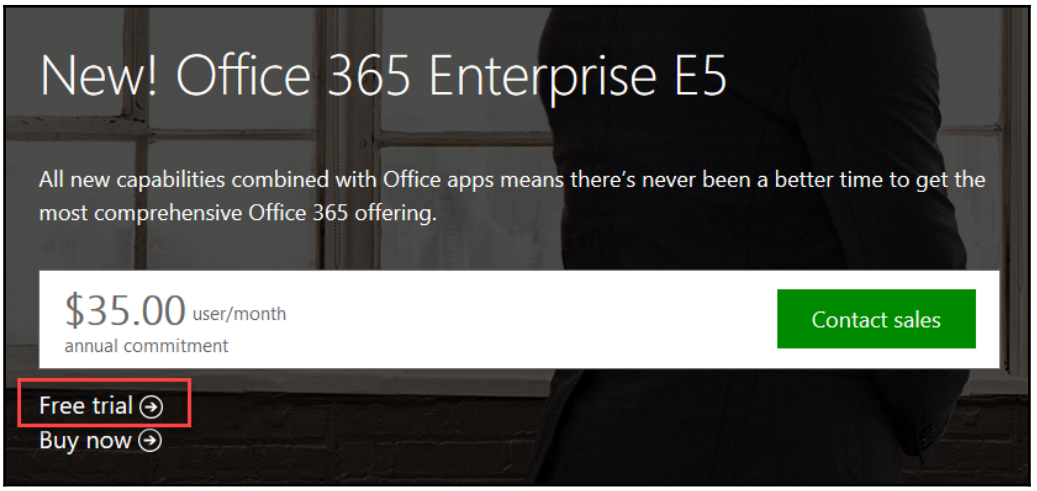
Chapter scenario overview

In the next section, we will start the configuration of the scenarios.

Implementing a solid Azure Active Directory

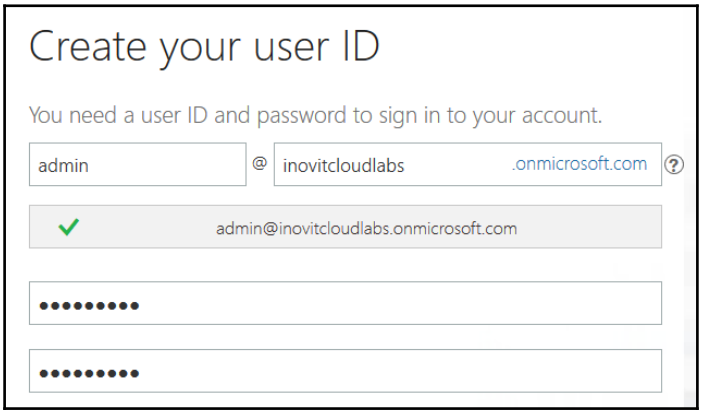
The first step we need to take is to get an Azure AD tenant. There are many ways to do this. You can start with an Azure subscription or use any other service from the Microsoft SaaS portfolio. The easiest way to get your solution to a working state is to start with an Office 365 trial subscription.

Open your browser and navigate to `http://bit.ly/1RVpFXe`. Subscribe to a free **Office 365 Enterprise E5** plan:



Office 365 E5 trial request

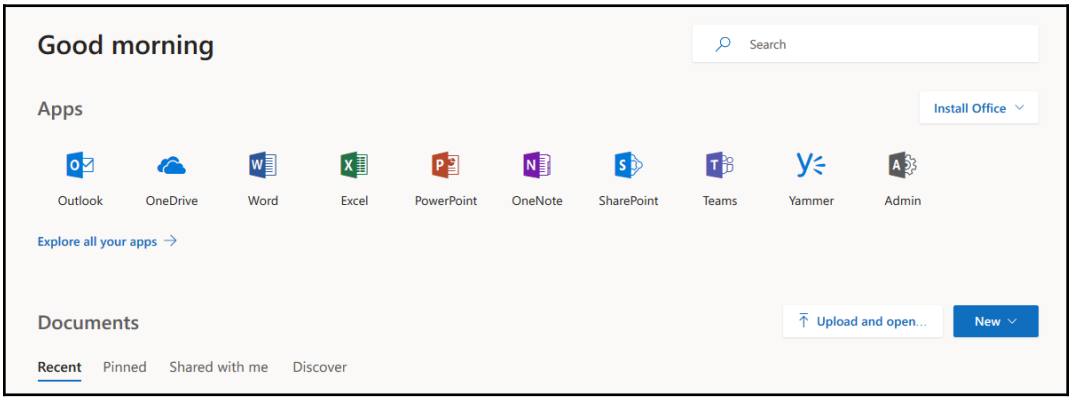
Follow the registration process and define your user ID, such as `admin@domain.onmicrosoft.com`. We recommend using a nonpersonal ID, as shown in the next screenshot. Enter your new user ID and password. Your default directory will get the name you define behind the `@`:



First Global Administrator creation

Afterward, you need to prove your identity with a text message or a phone call and enter the received code. Next, you need to click **Create my account**. Keep in mind that the provisioning process takes a few minutes and should end with a success message.

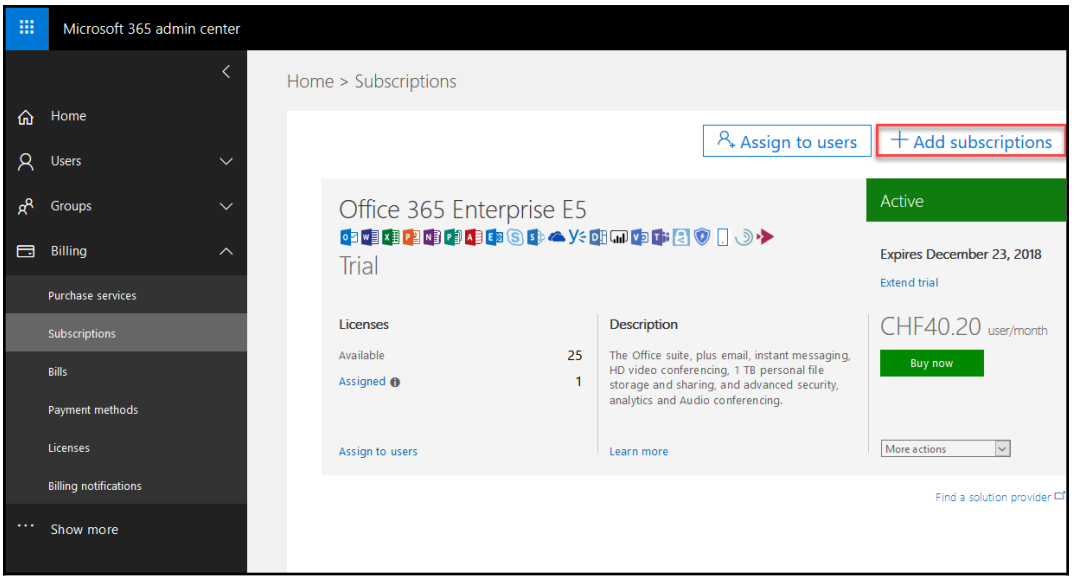
After the successful creation of your brand new Azure AD with an associated Office 365 E5 plan, you should be able to log on with your administrative credentials and see the following screen:



Office 365 management portal

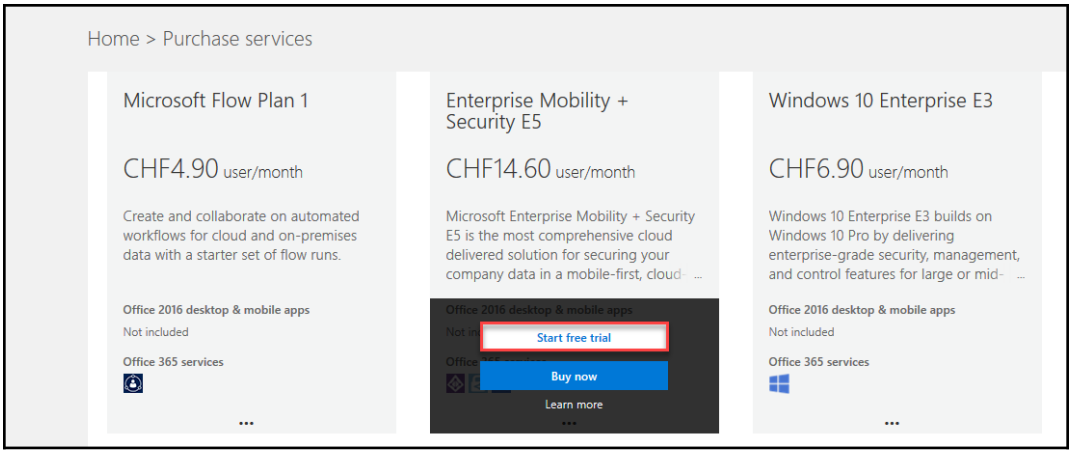
In the next step, we will assign an **Enterprise Mobility Suite** (**EMS**) E5 plan to the freshly created Azure AD tenant.

Click on the **Admin** icon on the right, and you should see your current assigned **Subscriptions** under the **Billing** tab:



Office 365 subscription management

Click **Add subscriptions** to add the EMS E5 trial plan to your Azure AD tenant:



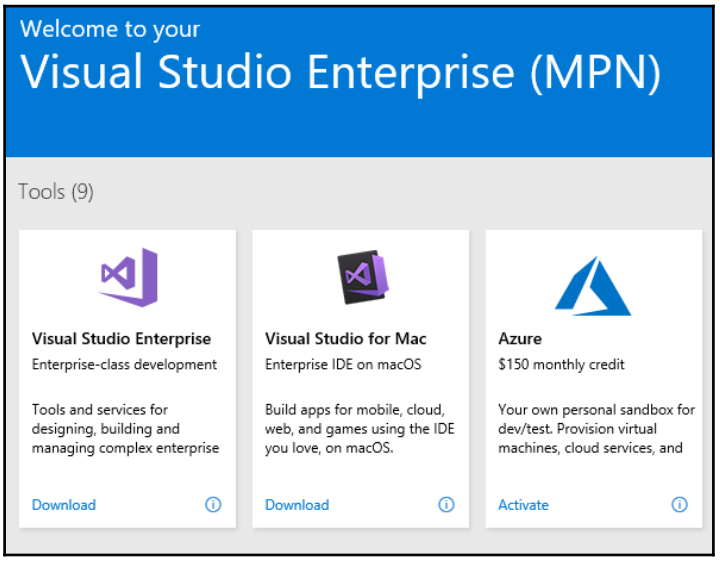
EM+S E5 trial request

Choose the EMS E5 plan and click **Start free trial** and follow the subscription process. After a successful subscription process, you can see the assigned Office 365 E5 and the EMS E5 plan in your Azure AD tenant.

Now that we have created our Azure AD tenant, we need to subscribe for an Azure free trial subscription. This step is necessary to use Azure resources such as the Azure AD Domain Services or other functionality we will discuss in the next chapters.

You can also use the following ways to get an Azure subscription:

- Use an Azure subscription from scratch (`https://account.azure.com/organization`)
- Use an agreement-based Azure subscription
- Use an MSDN Azure subscription, as shown in the following figure:



Visual Studio subscription benefits

Remember you can only sign up for one Azure AD free trial subscription.

Let's go to configure your administrative workstation and your personal Azure AD tenant.

Configuring your administrative workstation

First of all, we need to set a functional administrative workstation to work through this guide. You need to have a Windows 10 Enterprise client machine in a work group configuration. We recommend using a freshly installed Windows 10 Enterprise virtual machine. We need a Windows 10 device to use the Azure AD Join later in the book. If you are not able to access the Volume Licensed or MSDN version, you can use the Enterprise Evaluation version at `https://www.microsoft.com/en-gb/evalcenter/evaluate-windows-10-enterprise`.

In the code section of this chapter, you will find the following cmdlets to install the needed administrative tools on your client machine, basically, the Azure AD, MSOnline and the Azure Resource Manager PowerShell modules:

1. Install the Azure Active Directory PowerShell module:

```
Install-Module -Name AzureADPreview
```

2. Install the `MSOnline` PowerShell module:

```
Install-Module -Name MSOnline
```

3. Install the Azure Resource Manager PowerShell module:

```
Install-Module AzureRM
```

4. Connect to the `MSOnline` interface with PowerShell:

```
Connect-MsolService
# Provide your global administrator credentials
# View your assigned subscriptions
Get-MsolAccountSku
# View all actual users
Get-MsolUser
```

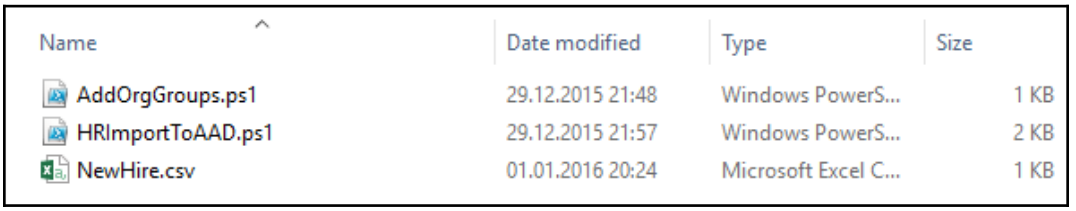
5. Create your first test user to prove the Azure AD administrative connection:

```
New-MsolUser -UserPrincipalName
"jochen.nickel@inovitcloudlabs.onmicrosoft.com" -DisplayName
"Jochen Nickel" -FirstName "Jochen" -LastName "Nickel" -
UsageLocation "CH" -LicenseAssignment
"inovitlabs:ENTERPRISEPREMIUM","inovitcloudlabs:EMSPREMIUM"

Get-MsolUser -UserPrincipalName
jochen.nickel@inovitcloudlabs.onmicrosoft.com | fl
```

6. Connect directly to the Azure AD interface to compare the output and capabilities with the `MSOnline` PowerShell module:

```
Connect-AzureAD
Get-AzureADUser -all $true | where userprincipalname -eq
jochen.nickel@inovitcloudlabs.onmicrosoft.com | fl
```

7. Unpack the deployment package from the code package. The `C:\Configuration\HRExports` directory contains the needed HR import and group creation scripts to configure your Azure AD tenant with some test data:



Name	Date modified	Type	Size
AddOrgGroups.ps1	29.12.2015 21:48	Windows PowerS...	1 KB
HRImportToAAD.ps1	29.12.2015 21:57	Windows PowerS...	2 KB
NewHire.csv	01.01.2016 20:24	Microsoft Excel C...	1 KB

Example script set

In the `HRImportToAAD.ps1` script, the following important variables will be used:

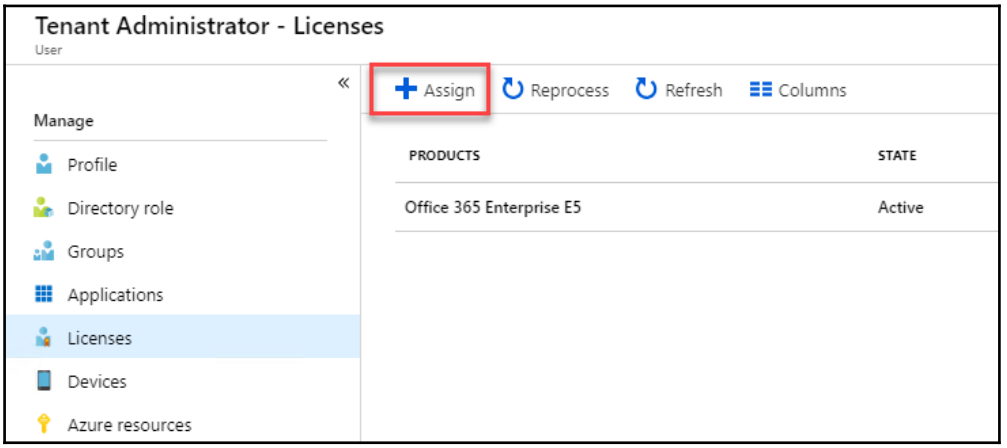
```
$domain = Get-MsolDomain | where {$_.Name -notlike "*mail*"}
$dir = "C:\Configuration\HRExports"

# Also configure your PowerShell Execution Policy to RemoteSigned with
the following cmdlet
# More information about this topic can be found under
http://bit.ly/1EWLG03
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
```

The domain variable will contain the `domain.onmicrosoft.com` name of your Azure AD default directory. We use this directory and not a registered domain name for different steps. At the end of the chapter, we will change to a custom domain so that you can explore the needed tasks. As you can see, the `dir` variable contains the path to the scripts and the simple HR export file called `NewHire.csv`. The `contoso.com` domain in the file will be replaced with your domain name, stored in the domain variable.

The `NewHire.csv` file contains the following demo user set, which will be used in future configurations to demonstrate the different functionalities:

```
userPrincipalName,DisplayName,FirstName,LastName,password
Don.Hall@contoso.com,Don Hall,Don,Hall,Pass@word1
Ellen.Adams@contoso.com,Ellen Adams,Ellen,Adams,Pass@word1
Jeff.Simpson@contoso.com,Jeff Simpson,Jeff,Simpson,Pass@word1
Brian.Cox@contoso.com,Brian Cox,Brian,Cox,Pass@word1
Doris.Sutton@contoso.com,Doris Sutton,Doris,Sutton,Pass@word1
Petro.Mitchell@contoso.com,Petro Mitchell,Petro,Mitchell,Pass@word1
```

With the next step, we will assign an EMS E5 plan license to our global administrator, `admin@domain.onmicrosoft.com`. The Office 365 E5 was already assigned through the creation process. Later in the chapter, we will assign licenses through dynamic group membership, which is an Azure AD Premium P2 license feature:



License assignment operation

Click **Assign** and add the EMS E5 plan license to your global administrator. The expected result will be as follows:

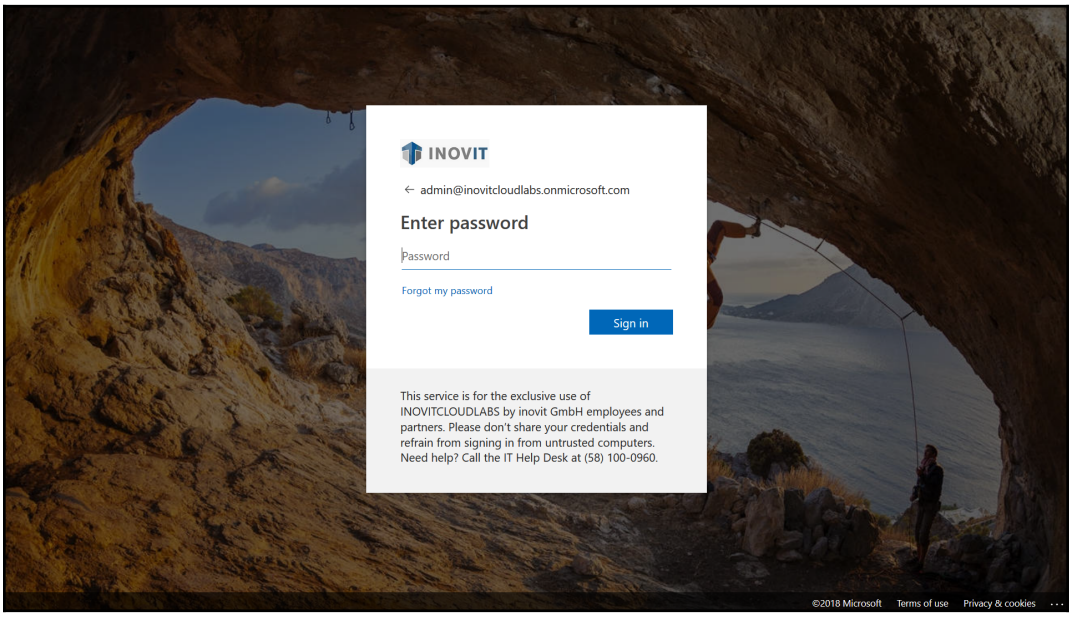
PRODUCTS	STATE	ENABLED SERVICES	ASSIGNMENT PATHS
Enterprise Mobility + Security E5	Active	9/9	Direct
Office 365 Enterprise E5	Active	25/25	Direct

Assigned license overview

We will get the correct message that we have no active subscription assigned to this user ID. Next, sign up for a Microsoft Azure subscription.

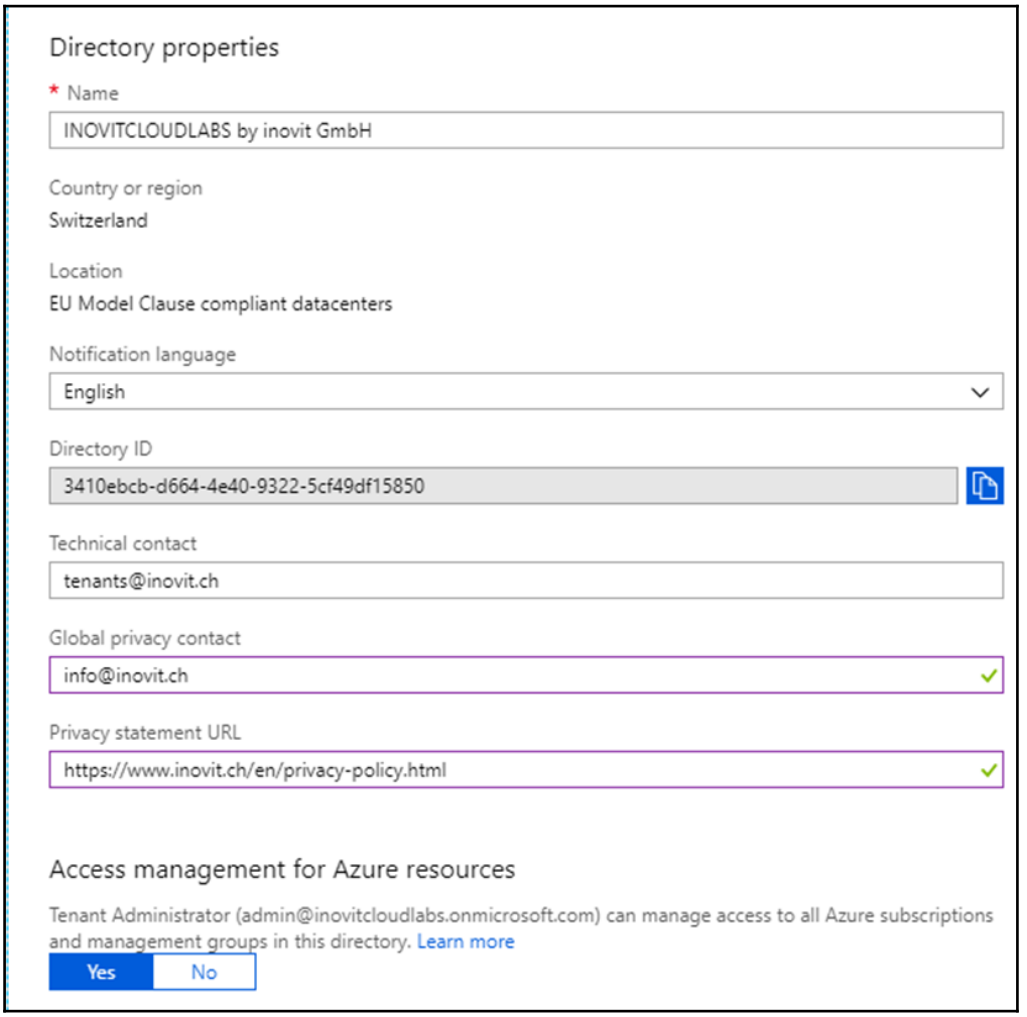
Custom company branding

Most companies like to see how they can apply their corporate identity to Azure services. With a few easy steps, you can show the most important capabilities. To add custom branding, you need to use an Azure Active Directory Premium 1, Premium 2, Basic, or Office 365 license. With the following simple example, you can see what you can customize. You can provide the customizing in different languages to address your own or your customers' needs. These configuration tasks are always a good starting point in a demo or a proof of concept. You are free to use your pictures and designs for this setup:



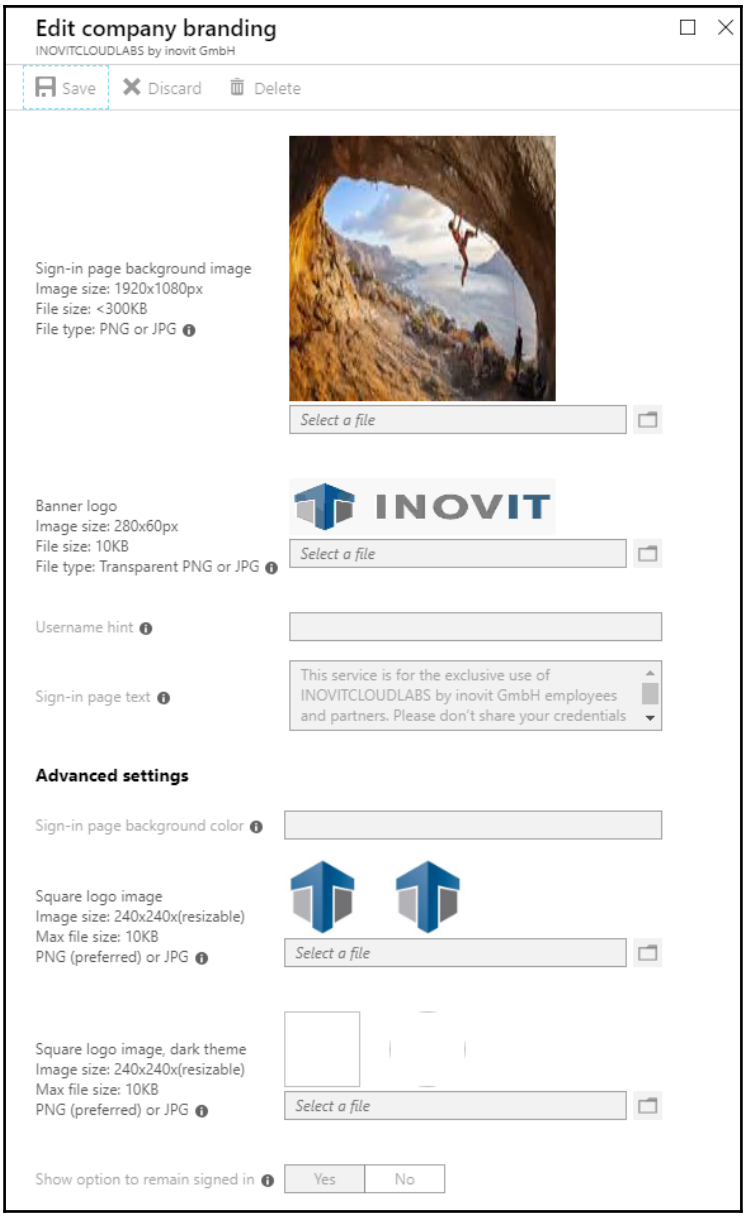
Customized portal example

The first thing we are going to change is the **Name** of the directory in the properties section. Just enter your desired name. We used `INOVITCLOUDLABS by inovit GmbH`. You can also provide your own technical and privacy contacts and links on the login page:



Azure AD tenant properties

Click **Customize Branding,** and you will see the following options. So that you can prepare your pictures and brands, we summarized the help information provided in Microsoft TechNet:



Azure AD portal-customizing options

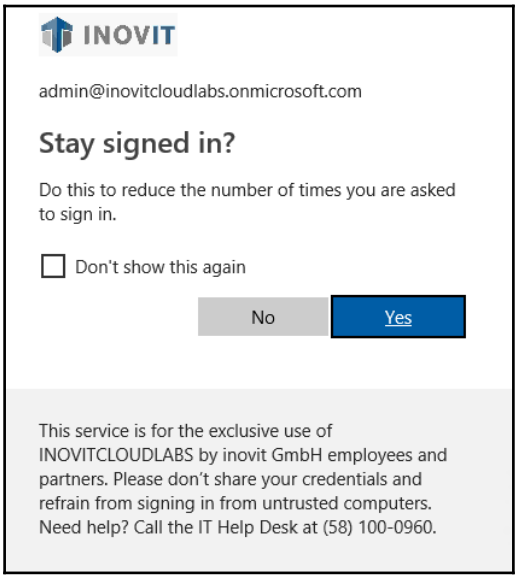
Next, you will see a configuration summary.

Summary and recommendations of the help information

The following section provides you with several capabilities and summarizes the most important corporate identity features to customize your environment:

- **Banner logo**: Choose between the following options:
  - Displayed on the Azure AD sign-in page and `myapps.microsoft.com`
  - PNG or JPEG
  - Can't be taller than 36 pixels or more extensive than 245 pixels
  - Recommendation—no padding around the image
- **Sign-in page text body**: Choose between the following options:
  - Appears at the bottom of the Azure AD sign-in page
  - Unicode text only with a maximum length of 256 characters
  - Use to communicate the phone number to your help desk or include a legal statement
  - Recommendation—don't add links or HTML tags
- **Sign-in page background image**: Choose from the following options:
  - Displayed on the side of the Azure AD sign-in page
  - PNG or JPEG
  - Recommended 1420 x 1200 with a supported file size of 300 KB (max. 500 KB)
  - Keep the exciting part in the top-left corner (image gets resized and cropped)
- **Username hint**: Hint text that appears to users if they forget their username:
  - Unicode, without links or code
  - Maximum 64 characters

- **Show option to remain signed in**: Let your users remain signed in to Azure AD until explicitly signing out:



Login experience

You are also able to do some extensive customization with the help of the following article `https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/customize-branding`.

Your expected result should be this:

Portal-customizing effect

Now that we have provided an essential company branding, we can start to create and manage users and groups.

Creating and managing users and groups

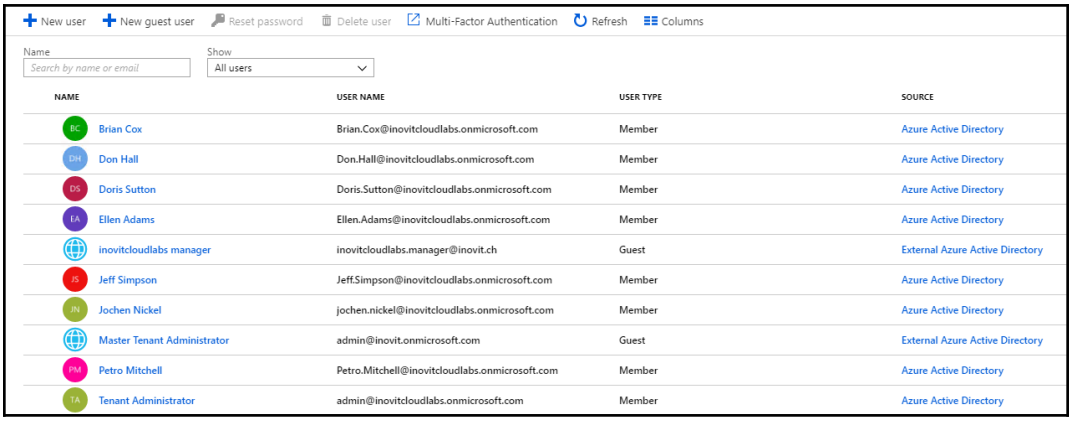
In the next steps, we connect to our Azure AD and generate the test users and groups.

Start the Azure AD PowerShell console and connect to Azure AD by executing the following cmdlets and scripts:

```
$msolcred = get-credential
# Enter your global administrator credentials
connect-msolservice -credential $msolcred
C:\Configuration\HRExports\HRImportToAAD.ps1
```

Alternatively, you can also use `connect-msolservice` directly to connect without the use of a variable.

After starting the script, go directly to `https://portal.azure.com` with your `admin@domain.onmicrosoft.com` credentials. Select the users' section under your Azure AD. You should find the users from the `HireUsers.csv` file under the **All users** tab:



Azure AD portal user management

Open `https://portal.office.com` | **Admin** | **Active Users,** and you can see your users with active licenses in Office 365:

☐	Brian Cox	Brian.Cox@inovitcloudlabs.onmicrosoft.com	Office 365 Enterprise E5 Enterp...
☐	Don Hall	Don.Hall@inovitcloudlabs.onmicrosoft.com	Office 365 Enterprise E5 Enterp...
☐	Doris Sutton	Doris.Sutton@inovitcloudlabs.onmicrosoft.com	Office 365 Enterprise E5 Enterp...
☐	Ellen Adams	Ellen.Adams@inovitcloudlabs.onmicrosoft.com	Office 365 Enterprise E5 Enterp...
☐	Jeff Simpson	Jeff.Simpson@inovitcloudlabs.onmicrosoft.com	Office 365 Enterprise E5 Enterp...
☐	Jochen Nickel	jochen.nickel@inovitcloudlabs.onmicrosoft.com	Office 365 Enterprise E5 Enterp...
☐	Petro Mitchell	Petro.Mitchell@inovitcloudlabs.onmicrosoft.com	Office 365 Enterprise E5 Enterp...
☐	Tenant Administrator	admin@inovitcloudlabs.onmicrosoft.com	Enterprise Mobility + Security ...

Office 365 user management

Let's create three example groups to represent the company organization with the following script:

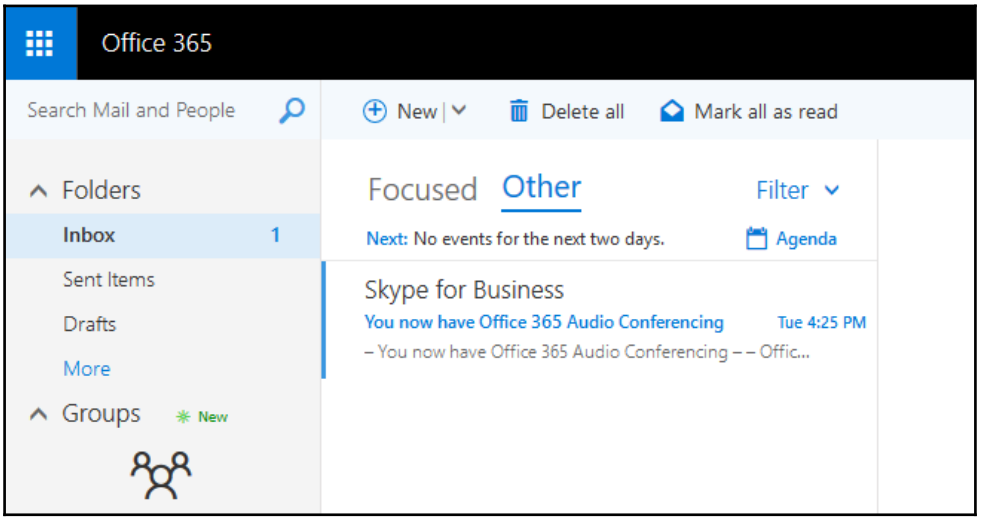
```
C:\Configuration\HRExports\AddOrgGroups.ps1
```

Now, you will see the created groups:

AC	Accounting	Security	Assigned
HR	HR	Security	Assigned
SA	Sales	Security	Assigned

Azure AD group management

Test your configuration, open `https://myapps.microsoft.com`, and log in with the user `Don.Hall@domain.onmicrosoft.com`, and you should see Office 365 SharePoint, Outlook, and many applications in the access panel UI. Click **Outlook,** and you should be able to open the app without additional login information to access your mailbox:



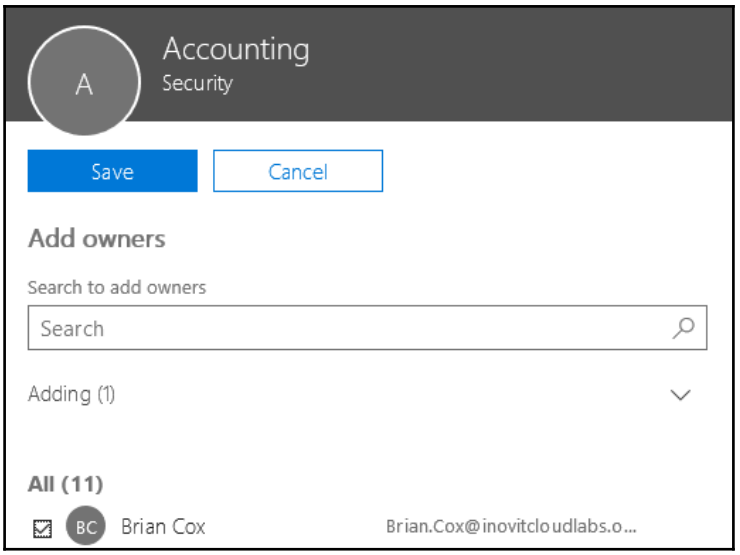
User Inbox dialog

In the next steps, we provide an owner to our organizational groups.

Set group owners for organizational groups

To provide group management by the manager of a department, we will assign the following users as owners of their department groups:

- **Accounting**: `Brian.Cox@domain.onmicrosoft.com`



Group - user assignment

Do the same for:

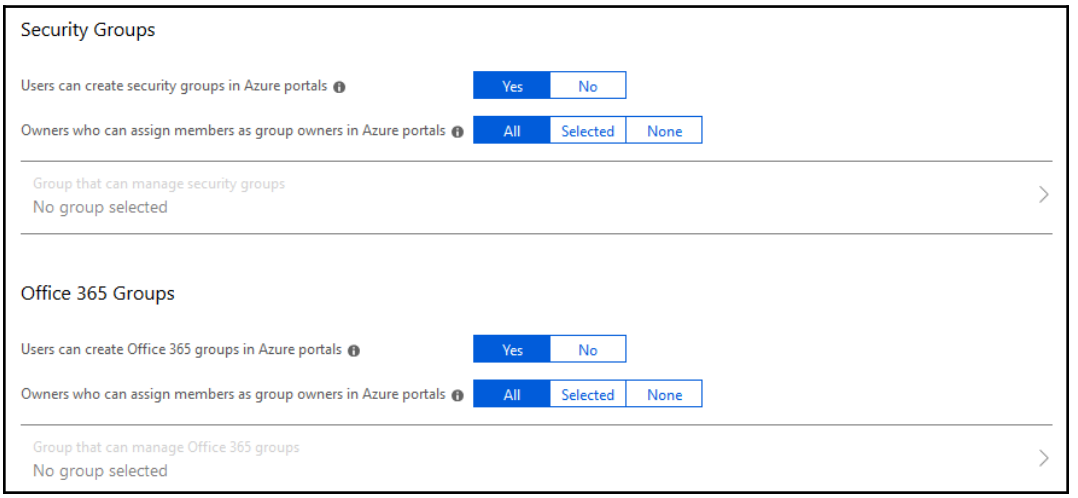
- **HR**: `Don.Hall@domain.onmicrosoft.com`
- **Sales**: `Doris.Sutton@domain.onmicrosoft.com`

Now that we have configured the owners, we will start to delegate management.

Delegated group management for organizational groups

The default configuration of Azure AD allows an owner of a security or Office 365 group to manage the group members based on the data owner concept in the Azure AD Access Panel and the Azure portal.

Furthermore, you can limit this functionality, based on your needs:



Group options in Azure AD

Log in as `Don.Hall@domain.onmicrosoft.com` to `https://myapps.microsoft.com`. Click on the **HR** group and add `Ellen.Adams@domain.onmicrosoft.com` to the **HR** group:

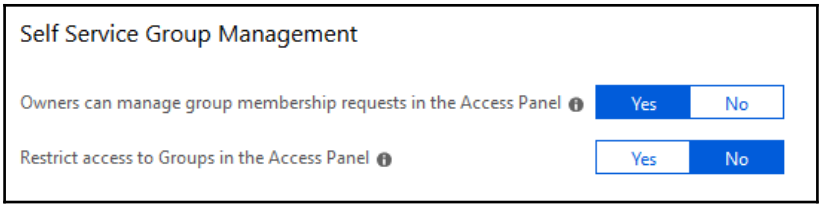
Group view in Azure AD access panel UI

Review the **Join policy** under **Edit details**.

In the next section, we will configure the group self-service options.

Configure self-service group management

Another request may be that users need to be able to create request-based security or Office 365 groups, for instance for projects or distribution groups. For this, they need the capability of an approval process. You can provide this functionality by activating the option under the group management general section. The feature set requires Azure Active Directory Premium:



Self-Service Group Management options

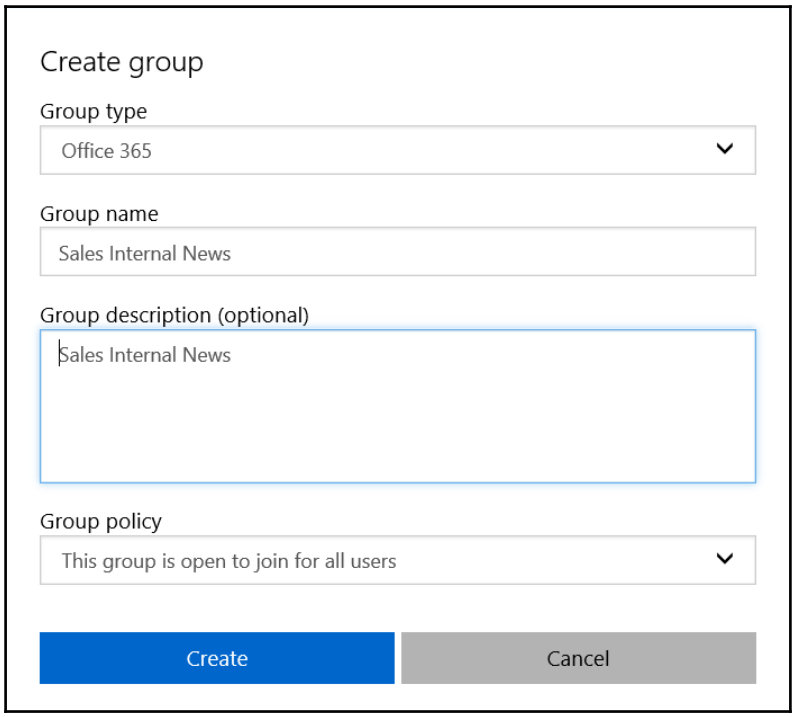
An Office 365 group includes a distribution list but also consists of these shared tools:

- Inbox for group email communication
- Calendar for scheduling group meetings and events
- Library for storing and working on group files and folders
- OneNote notebook for taking project and meeting notes
- Planning tool for organizing and assigning tasks and getting updates on project progress
- Guest access (set up by the administrator)

Practical note: Use a different browser or the **Private Browsing** option for handling the different user sessions: one session on `https://portal.azure.com` as `admin@domain.onmicrosoft.com` (Admin) and another session as the explicit user (User) under `https://myapps.microsoft.com`.

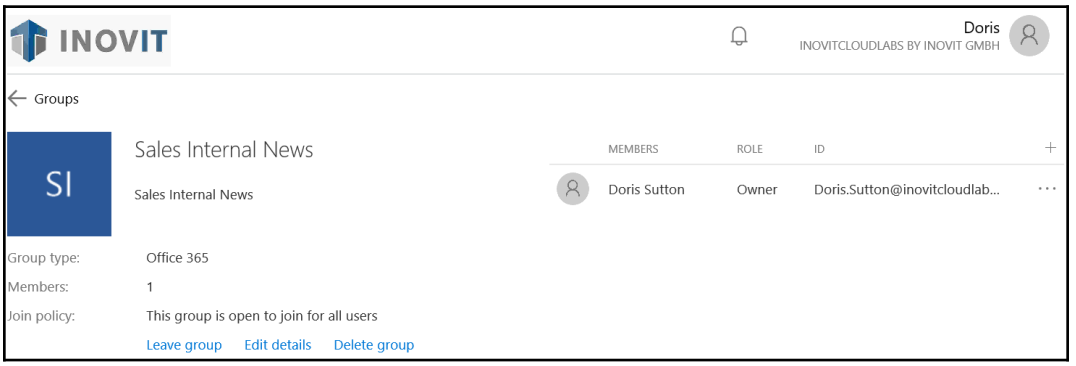
Create the sales internal news group as an Office 365 (distribution group)

Log in as `Doris.Sutton@domain.onmicrosoft.com` to `https://myapps.microsoft.com` and create the `Sales Internal News` group as an Office 365 group. Check that the **Group policy** shows **This group is open to join for all users**:



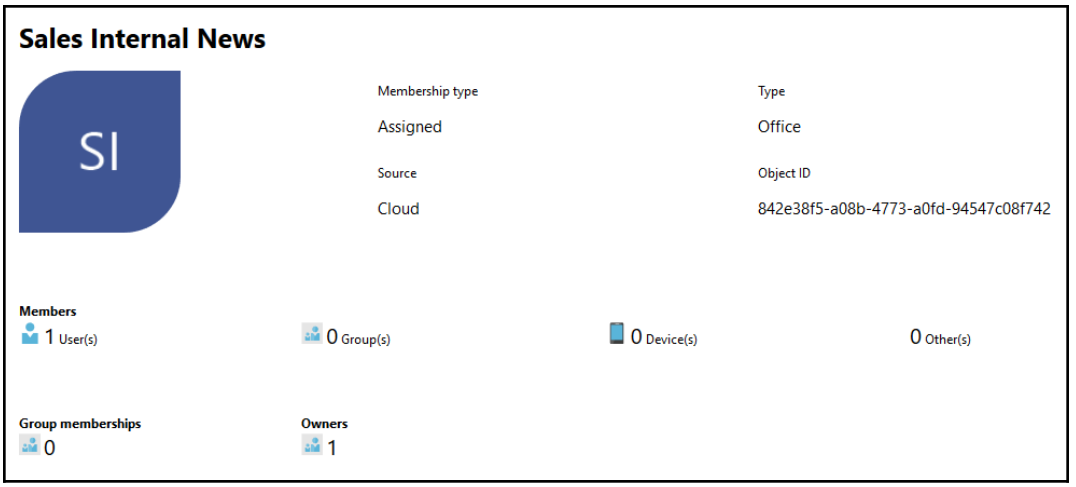
Azure AD access panel UI - group creation

Review the **Join policy** of your newly created group:



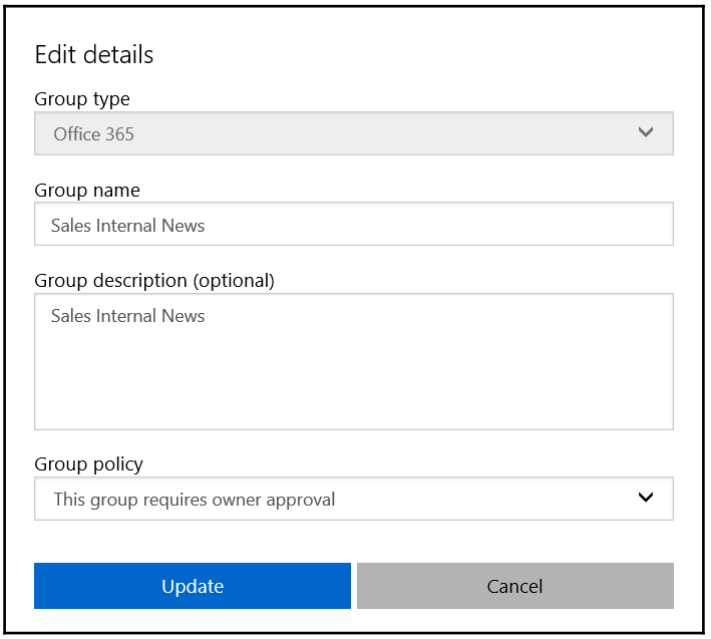
Group dialog - Azure AD access panel UI

In your Azure AD, under **Groups**, you will also find the newly created group:



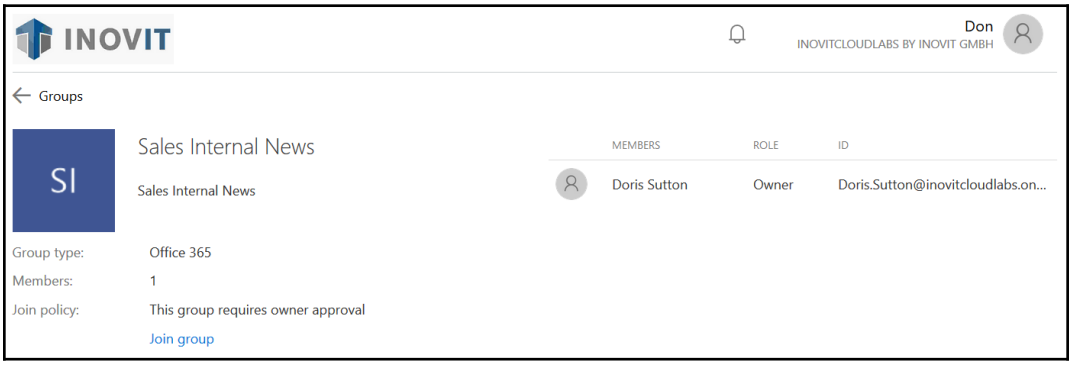
Group overview - Azure AD access panel UI

Now, as the group owner, we change the group to request a managers' approval with the group policy setting:



Group editing dialog

Test the new configuration and log in as `Don.Hall@domain.onmicrosoft.com` to `https://myapps.microsoft.com`. Navigate to groups. Choose **Sales Internal News**:



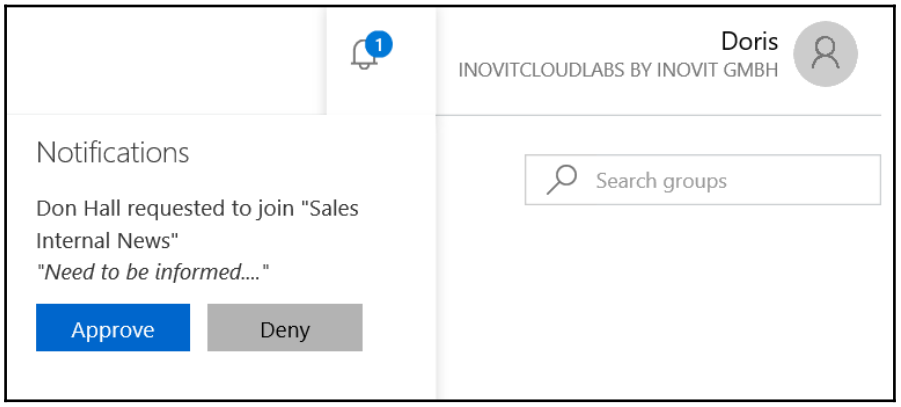
Join group dialog

Join the **Sales Internal News** group and type a `Business justification`, click **Request**, and the process should be started.

Log in as `Doris.Sutton@domain.onmicrosoft.com` to `https://myapps.microsoft.com`.

Check your inbox. You should have received the join request mail and a notification, shown in the Access Panel UI.

Click on this request and approve it:

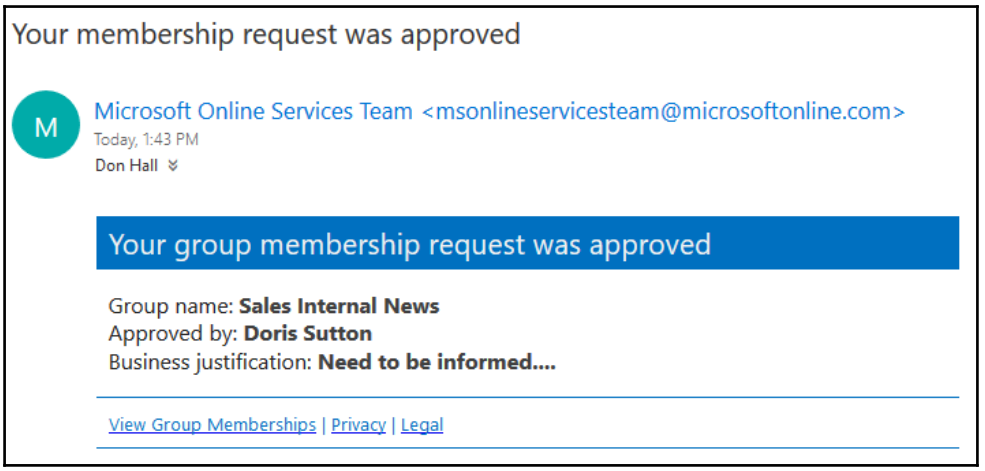


Group join - Notifications

Note: Next, you will see the group members of the **Sales Internal News** group.

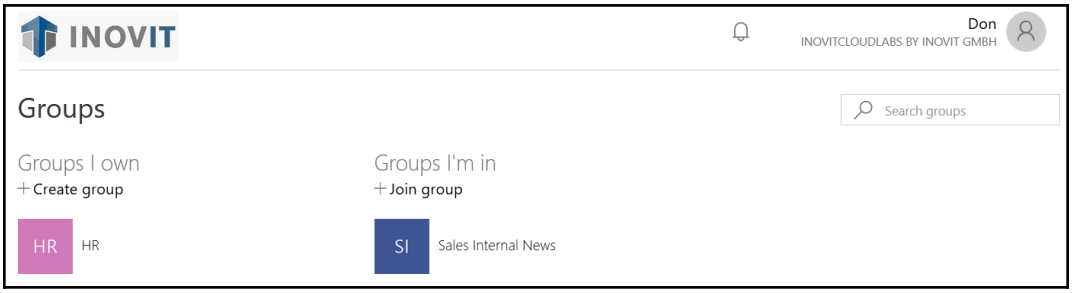
Log in as `Don.Hall@domain.onmicrosoft.com` on `https://myapps.microsoft.com`.

Check your inbox, and you should have received a successful approval message:



Approval message - group membership

Check your group membership, and you should be a member of the **Sales Internal News** group:



Group management in Azure AD access panel UI

Next, we will configure dynamic group memberships.

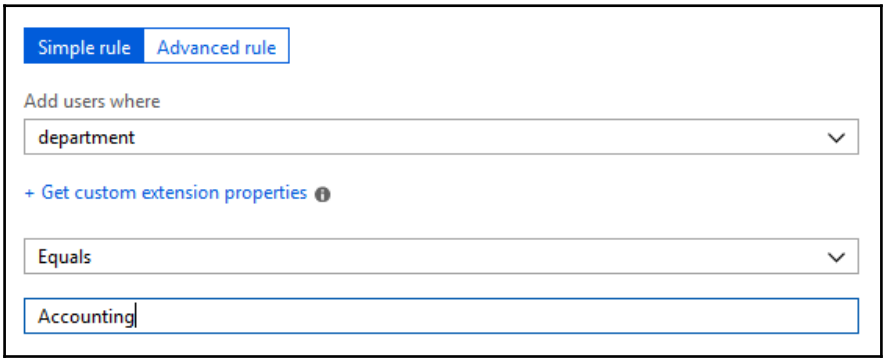
Configure dynamic group memberships

In the next section, we will configure straightforward dynamic group memberships to use the department attribute to add users to their department group and build up a dynamic licensing assignment. Group-based licensing currently does not support groups that contain other groups (nested groups).

An Azure AD Premium P1 license is needed for every user in a dynamic group.
When enabling dynamic groups, current memberships will be lost.
The usage location of a user needs to be set to assign a license.

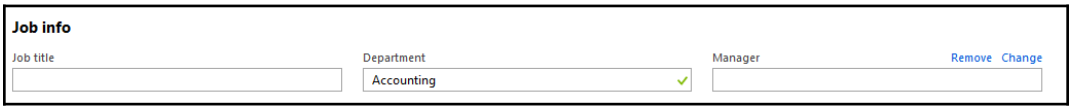
As the `admin@domain.onmicrosoft.com`, choose the **Accounting** group, navigate to properties, and change the membership type to **Dynamic User**.

Create a simple rule, department **Equals** (-eq) `Accounting`:



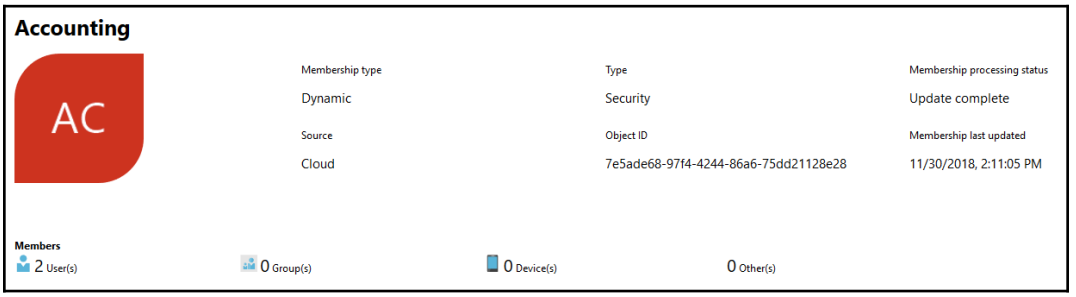
Dynamic group membership rule configuration

Set the department attribute (profile section) on the accounting users `Brian Cox` and `Jeff Simpson` to **Accounting**:



Filling user attributes for dynamic group usage

The member should be added automatically. Check the group membership and verify the two new members:

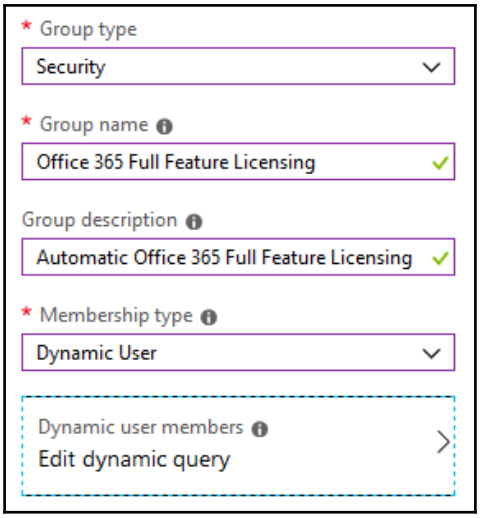


Freshly calculated dynamic group membership

Next, we will provide an automatic licensing solution.

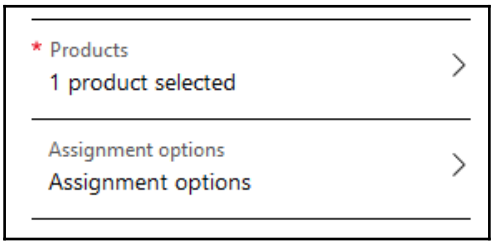
Create the following security group:

- Office 365 full feature licensing
- **Group description**: **Automatic Office 365 Full Feature Licensing**
- **Membership type**: **Dynamic User**
- **Dynamic query**: `userType -eq Member`:



Group properties dialog

Under **Licenses** | **Products**, assign the Office 365 E5 plan. Don't choose any assignment options at the moment:



Group assignment options

Note: With the assignment options, you can enable/disable features as needed.

Wait until the membership has updated and check the license assignment for `Don.Hall@domain.onmicrosoft.com`.

You will see that the user gets the license through a direct and group-based assignment:

PRODUCTS	STATE	ENABLED SERVICES	ASSIGNMENT PATHS
Enterprise Mobility + Security E5	Active	9/9	Direct
Office 365 Enterprise E5	Active	25/25	Direct, Inherited (Office 365 Full Feature Licensing)

License assignment overview

This license solution is to give you a starter. You should remove the directly assigned licenses from all users that get licenses from group membership.

In the next section, we will configure role assignments to administrative units.

Assign roles to administrative units

To delegate tasks, we use the creation of **administrative units** (**AUs**) and assign roles for specific tasks. In this configuration, we generate an `HR [AU]`, and we assign the manager of the HR department with the role to manage user accounts in this scope.

Creating an administrative unit

First of all, we need to connect to our Azure AD with the PowerShell cmdlet Connect-AzureAD for the `admin@domain.onmicrosoft.com` user.

Use the following cmdlets to create the `HR [AU]`:

```
New-AzureADAdministrativeUnit -Description "Human Resources Users" -
DisplayName "HR"
```

View the expected output:

```
ObjectId                             DisplayName Description
--------                             ----------- -----------
8e36f339-b7cc-4f18-ad4f-89188f6f1025 HR          Human Resources Users
```

Newly created administrative unit

Next, we will add the related users.

Adding users to an administrative unit

Next, we add the users of the HR department to the `HR [AU]`. Use the following cmdlets to do this:

```
$HRAU = Get-AzureADAdministrativeUnit -Filter "displayname eq 'HR'"
$initialDomain = (Get-AzureADDomain)[0].Name
$HRUser1 = Get-AzureADUser -Filter "UserPrincipalName eq
'don.hall@$InitialDomain'"
$HRUser2 = Get-AzureADUser -Filter "UserPrincipalName eq
'ellen.adams@$InitialDomain'"
Add-AzureADAdministrativeUnitMember -ObjectId $HRAU.ObjectId -
RefObjectId $HRUser1.ObjectId
Add-AzureADAdministrativeUnitMember -ObjectId $HRAU.ObjectId -
RefObjectId $HRUser2.ObjectId
Get-AzureADAdministrativeUnitMember -ObjectId $HRAU.ObjectId | Get-
AzureADUser
```

The output of the preceding command is as follows:

```
ObjectId                             DisplayName UserPrincipalName                          UserType
--------                             ----------- -----------------                          --------
44fb9355-0a23-4487-8059-206b62da16a6 Don Hall    Don.Hall@inovitcloudlabs.onmicrosoft.com    Member
0b242043-acdf-4c8a-8b26-f03eb913cbd7 Ellen Adams Ellen.Adams@inovitcloudlabs.onmicrosoft.com Member
```

Newly added users overview

Next, we will use the scoping options.

Scoping administrative roles

In the next step, we assign the user account administrator role. Verify available roles with the following cmdlet:

```
Get-AzureADDirectoryRoleTemplate
```

Now, we enable the user account administrator role with the following cmdlet:

```
Enable-AzureADDirectoryRole -RoleTemplateId
fe930be7-5e62-47db-91af-98c3a49a38b1
```

Set variables and assign the user to the role:

```
$admins = Get-AzureADDirectoryRole
foreach($i in $admins) {
    if($i.DisplayName -eq "User Account Administrator") {
        $uaAdmin = $i
       }
    }

$HRUA = Get-AzureADUser -Filter "UserPrincipalName eq
'Don.Hall@$InitialDomain'"
$uaRoleMemberInfo = New-Object -TypeName
Microsoft.Open.AzureAD.Model.RoleMemberInfo -Property @{ ObjectId =
$HRUA.ObjectId }
Add-AzureADScopedRoleMembership -RoleObjectId $uaAdmin.ObjectId -
ObjectId $HRAU.ObjectId -RoleMemberInfo $uaRoleMemberInfo
```

The output of the preceding command is as follows:

```
ObjectId                             DisplayName              Description
--------                             -----------              -----------
dd0bdb75-7631-49ca-be3f-6831d7428c41 User Account Administrator Can manage all aspects of users and groups, includin...
```

User Account Administrator assignment

Next, we will test our configuration.

Test your configuration

Open a new PowerShell and connect with the `Connect-MsolService` command to the Azure AD, and log in with `Don.Hall@domain.onmicrosoft.com` credentials.

Modify a user account assigned to the HR administrative unit:

```
Set-MsolUser -UserPrincipalName ellen.adams@domain.onmicrosoft.com -
Department HR
```

Verify your modification:

```
Get-MsolUser -UserPrincipalName ellen.adams@domain.onmicrosoft.com |
select Department
```

Next, we will protect an administrative account with the **Privileged Identity Management** (**PIM**) features of Azure AD Premium P2. We recommend using Azure MFA to protect your administrative accounts, if you don't want to invest in Azure AD Premium P2.

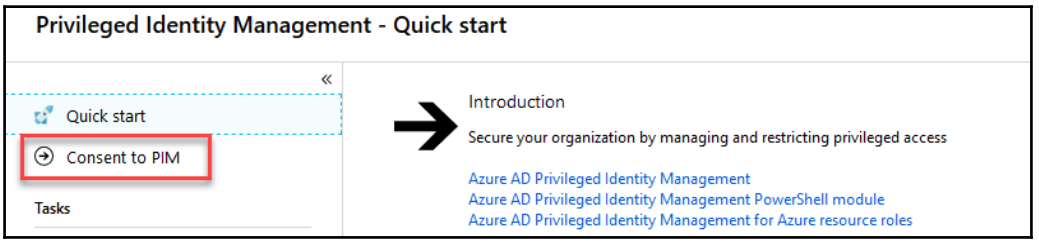
Protect your administrative accounts

In this section, we will use Azure AD Premium P2 PIM to protect an administrative account in a quick intro.

Open `https://portal.azure.com` as `admin@domain.onmicrosoft.com` to start the configuration.

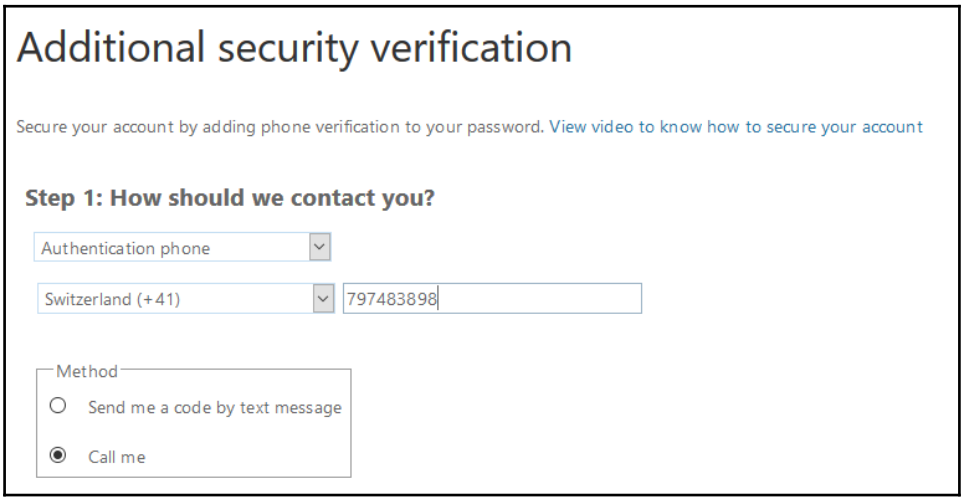
Click **All Services** and choose the Azure AD Privileged Identity Management.

Now, we need to **Consent to PIM** to use the service:



Privileged Identity Management - enablement

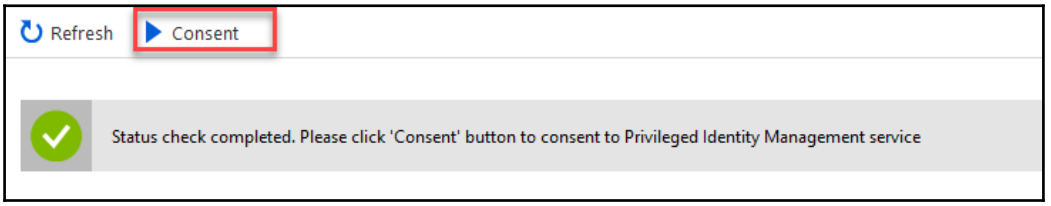
You will need to verify your identity and provide your preferred security verification option, as you can see in the following screenshot:



Azure MFA onboarding

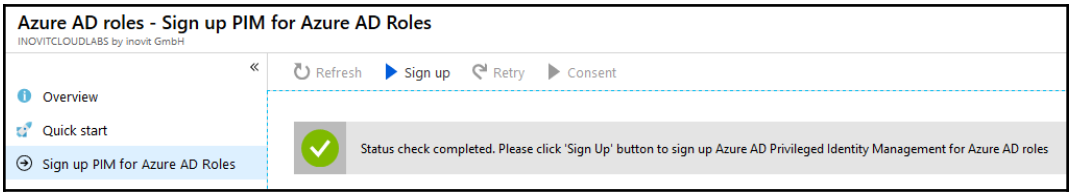
If you already use the Microsoft Authenticator App on your mobile device, you can also register the mobile app.

Finish the verification process and click **Consent**—proceed:



Consent to finish the initialization

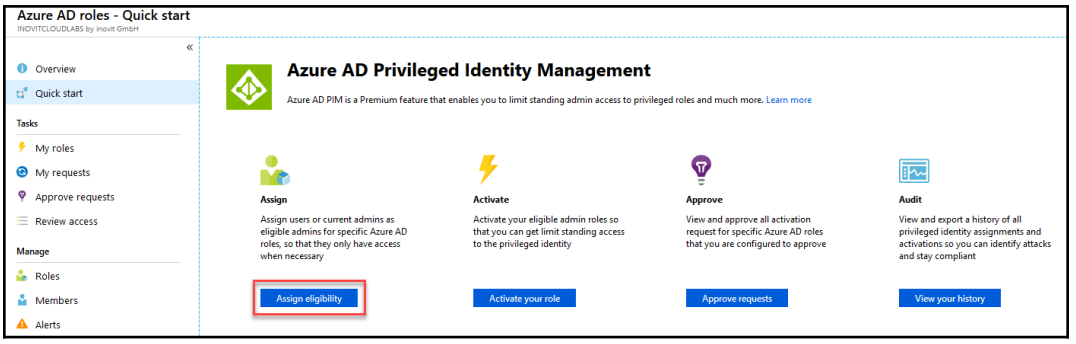
Next, we sign up under Azure AD Roles, so that users can enable Azure AD roles. Click **Sign up PIM for Azure AD Roles** to activate the functionality:



Azure AD roles - PIM sign-up

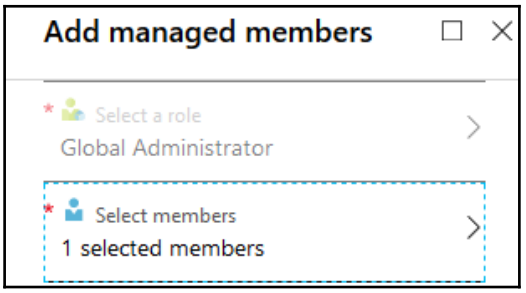
Now that the feature is enabled, we can assign the roles to our users.

Click **Assign eligibility** to start the task:



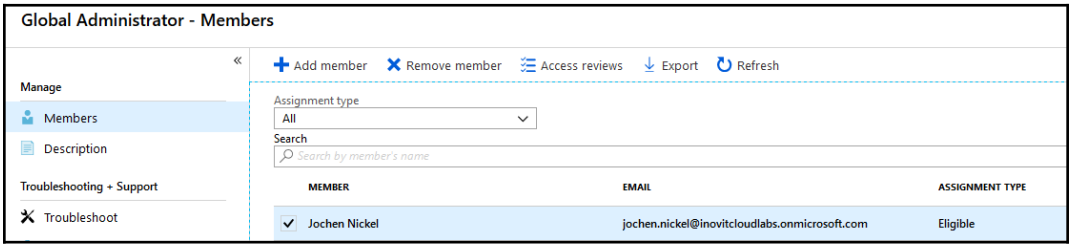
Role assignment procedure

Click the **Global Administrator Role**, view the actual members, and add your test account to the role:



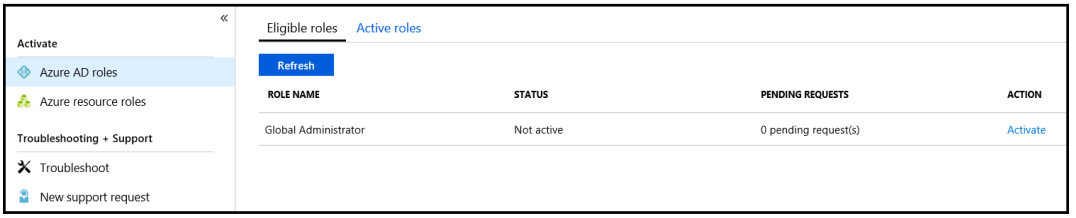
User assignment to a role

View the expected result:



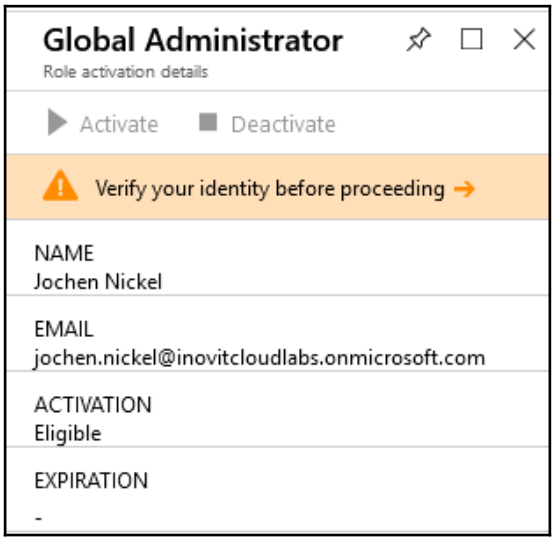
New eligible user assigned to the role

Let's test our configuration by opening an `InPrivate` browser session; open `https://portal.azure.com` and log in with your own test account. Click **All Services** and choose **Azure AD Privileged Identity Management**. Choose **My roles** and activate the **Global Administrator** role for your account:



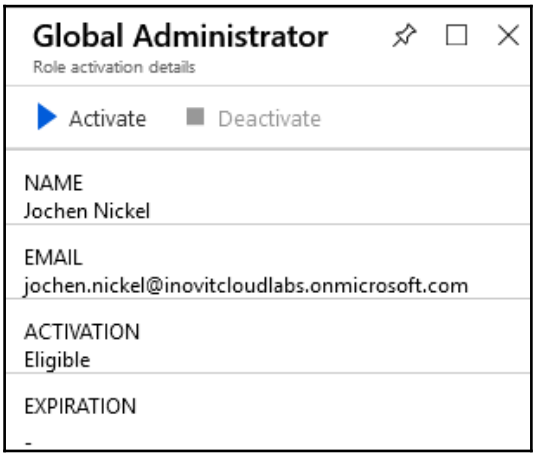
Role activation procedure

Next, you need to verify your identity. Follow the process, register, and verify your account. You need to complete the registration process just once:



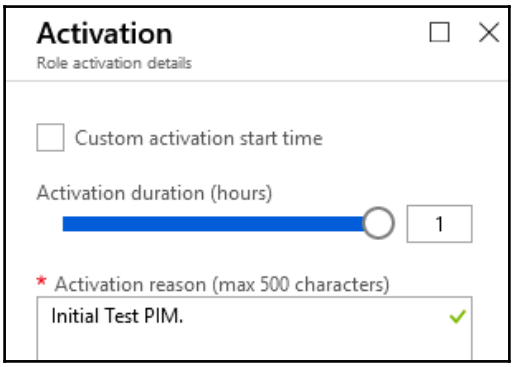
Starting the verification process

After the registration and verification processes are finished, you can **Activate** your role:



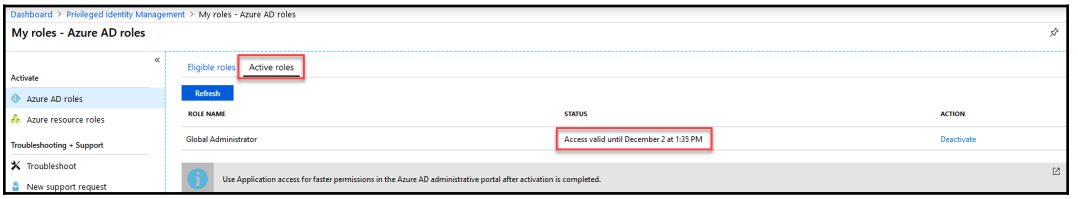
Role activation

Provide a reason for your role activation. You will note that the role is limited for `1` hour and that you can define a custom activation time. Later in the book, we will configure different roles and features:



Activation options, such as Custom activation start time and Activation reason

Verify that your role is activated. You have successfully requested your **Global Administrator** role for the first time over **Azure AD** PIM. This is very useful so that high privileges are not permanently assigned to your account:



Active roles overview

We always recommend that you leave one **Global Administrator** permanently assigned, and that no Azure MFA is required to use the account. Use this account as a **Breaking Glass** account if the Azure AD PIM or MFA service is not available.

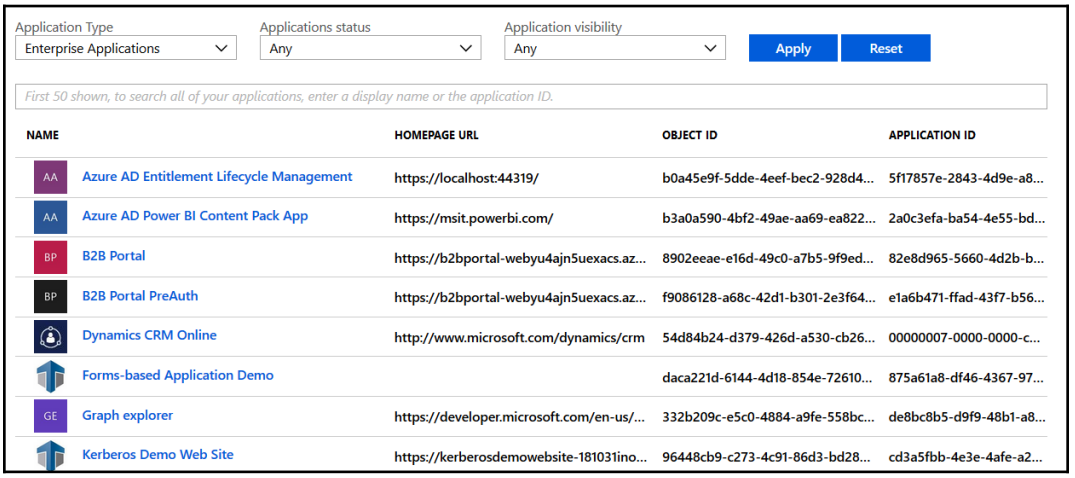
Next, we will configure user and group-based application access in Azure AD.

Provide user and group-based application access

In this section, we configure a typical workplace, which a user can access under the Access Panel UI (`https://myapps.microsoft.com`). We assign applications to users and groups to see the different capabilities. The steps don't contain all single sign-on or provisioning options. We will discuss these feature sets later in specific chapters.

Log in to `https://portal.azure.com` with your **Global Administrator** credentials and add several applications from the application gallery under the **Enterprise applications** section. After adding the applications, we assign the accounts, which are to be provided access.

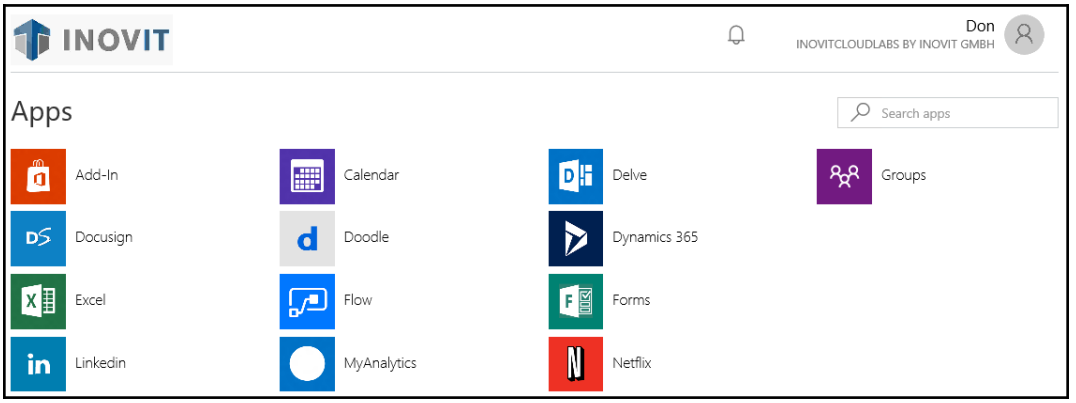
Build a list of applications like the following, and assign all groups to access the applications, except the one with user provisioning:



Azure AD application management

You will note the differences in the format with and without user provisioning.

Test your newly configured workplace and log in as `don.hall@domain.onmicrosoft.com` to `https://myapps.microsoft.com`:



Azure AD access panel UI - application access

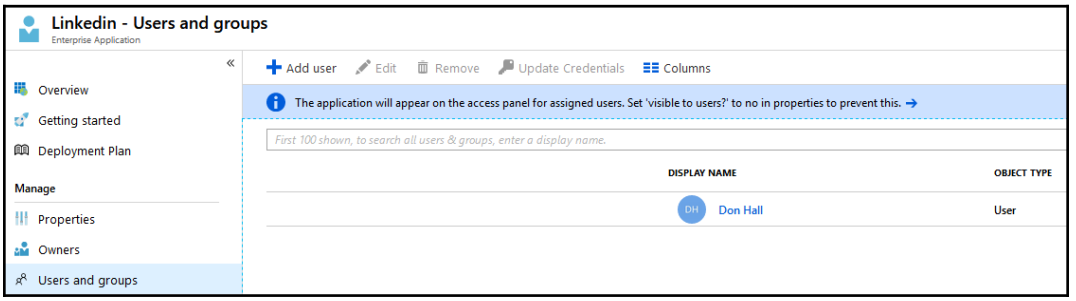
Also, test the user experience on Office 365 and log in as `don.hall@domain.onmicrosoft.com` to `https://portal.office.com`.

Next, we will assign applications to users.

Assign applications to users and define login information

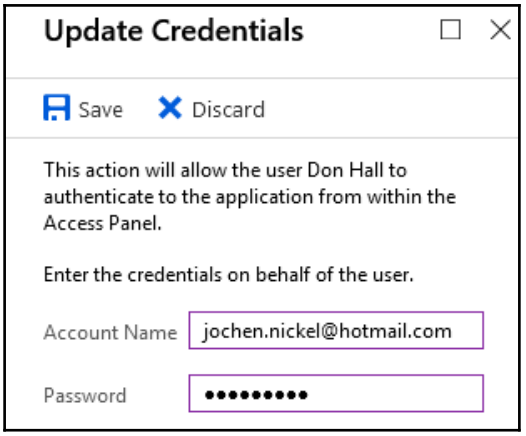
In the next step, we will assign the LinkedIn application to Don Hall, with the company credentials. Don Hall will not be able to see the credentials. So, if he leaves the company, the credentials are still protected.

Add the LinkedIn application from the application gallery and assign Don Hall to access this application:



Application - Users and Groups assignment

Next, we provide valid LinkedIn credentials. If you don't have a LinkedIn account, register a demo one:



Azure AD Credentials store

If you assign an application to a group, you can decide whether the credentials will be shared:

The following options for working with credential sets are available:

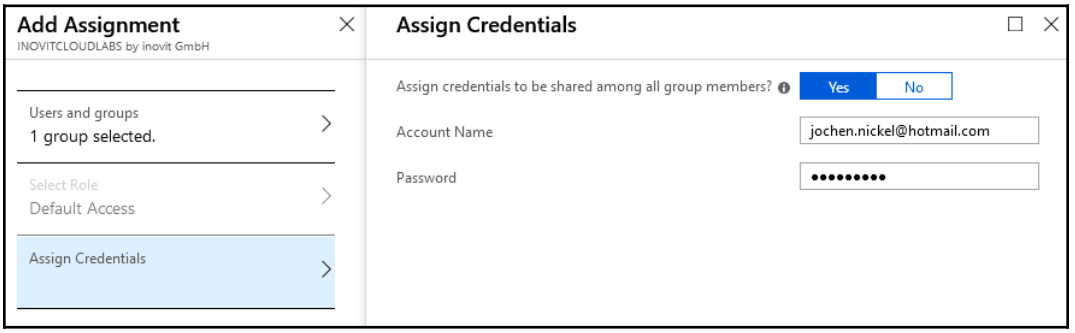
- **View with shared credentials**: Users can view this just a few seconds after clicking the application. Test it with the Twitter app.
- **View with no shared credentials**: Users need to add the preferred credentials once. Test it with the Twitter app.

You get the same behavior if you, as the administrator, don't provide the credentials.

Test this behavior with the user account, assign applications to groups, and define login information.

Assign applications to groups and define login information

In the next steps, we do the same for groups, such as an HR group that uses these groups to get news from an individual Twitter channel:



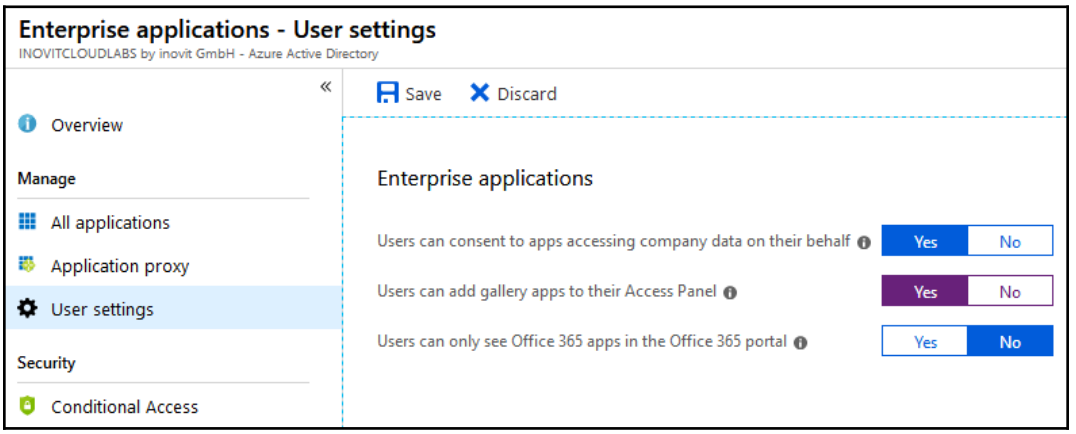
Filling the Azure AD Credentials store for application access

Add the Twitter app from the application gallery. Assign the HR group and configure the credentials. Check the application at `https://myapps.microsoft.com` with a user of the HR group.

Now, we can configure the self-service application management.

Self-service application management

In this section, we allow users to add their applications to their workplace under `https://myapps.microsoft.com` to enjoy the **Single Sign-On** feature. Under your Azure AD in the **Configure** section, navigate to **Enterprise applications**. Activate the function shown here:



Enterprise applications management options

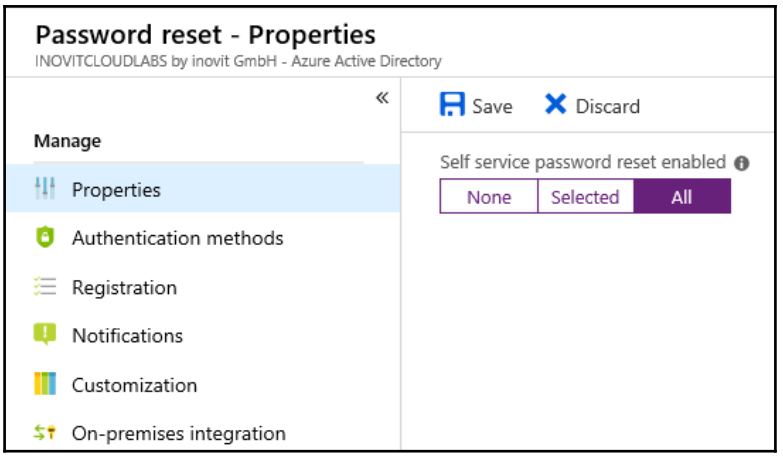
Log in as `Ellen.Adams@domain.onmicrosoft.com` to `https://myapps.microsoft.com`. Click **Get more applications** and add MailChimp. After you add MailChimp, click **Manage applications** and choose MailChimp.

As `admin@domain.onmicrosoft.com` on `https://manage.windowsazure.com`, you can see the newly added application under the **Enterprise applications** section of your Azure AD.

Configure the application, add some users, and test it!

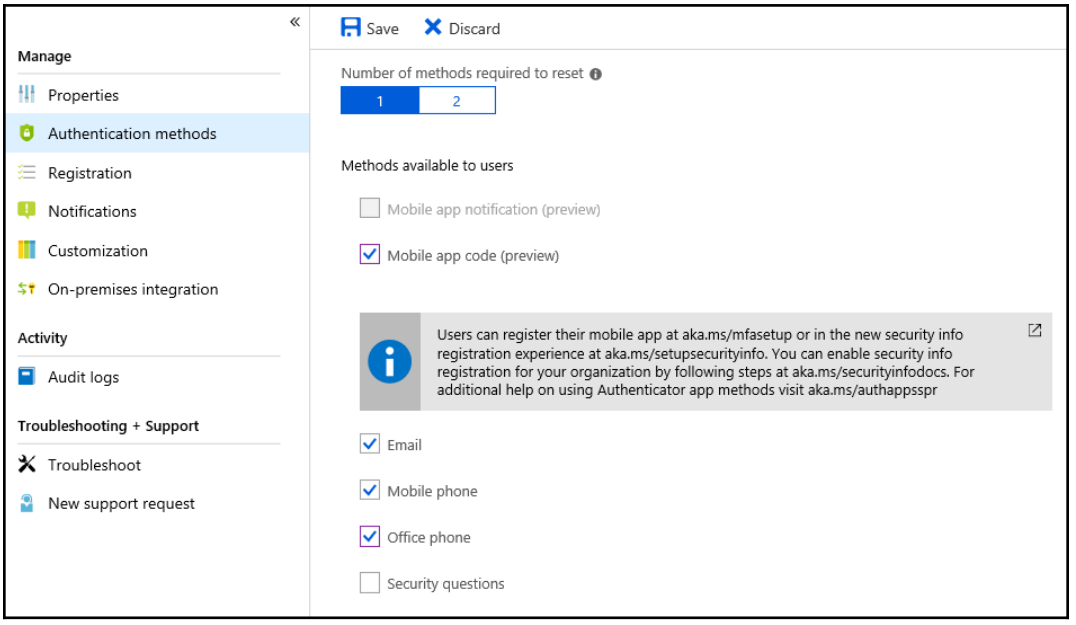
Password reset self-service capabilities

In this section, we configure the password reset capabilities of Azure AD to reduce support costs and 24/7 availability. We use no restrictions on the service and we require just one verification option to reset the password:



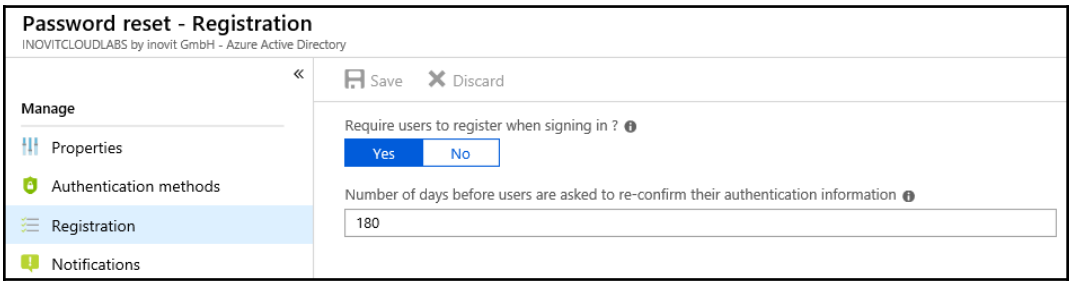
Password reset - Properties dialog to select the activation options

To verify the reset, we use several methods:



Password reset - authentication options

The next option we activate forces the user to register:



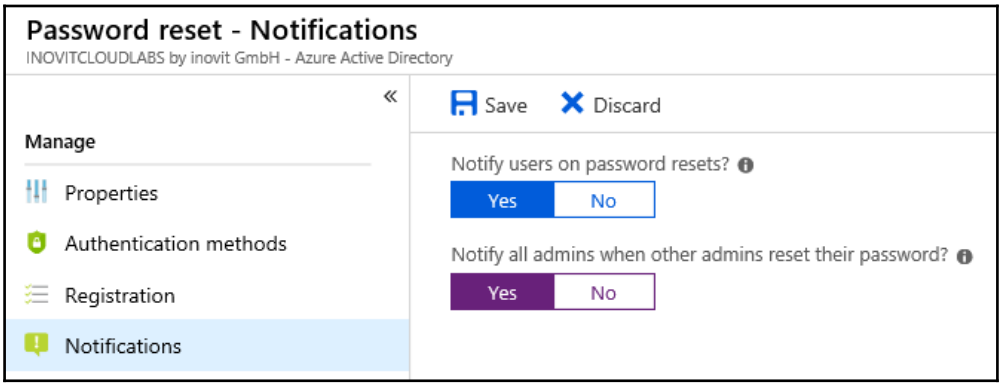
Password reset - Registration requirement and confirming choices

Next, we configure the related notifications.

Configure notifications

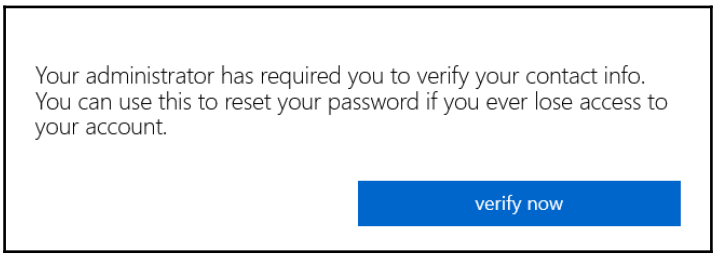
In this section, we configure the notifications options so that the administrator will be notified if anomalous sign-ins or administrator password resets happen:

1. Configure the **Notifications** as shown here:



Password reset - NotificationS options

2. Users will be forced to register for a password reset, as shown in the following screenshot:

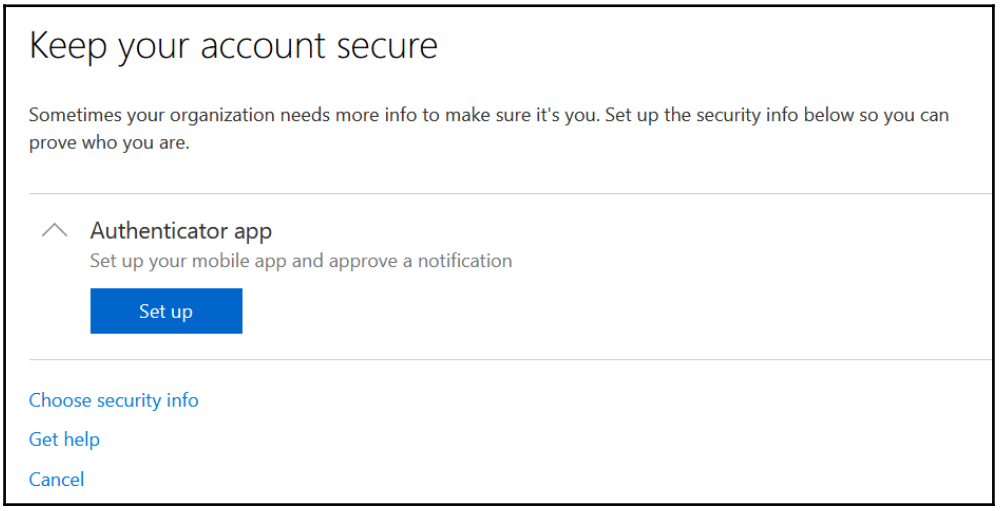


Registration enforcement

Now, we will test our newly configured feature and will see the registration scenario required for your verification options. Next, we will check the password reset.

Test the newly configured settings and log in as `Don.Hall@domain.onmicrosoft.com` to `https://myapps.microsoft.com`.

You will receive a message that you need to register for a password reset:



Authenticator app - setup procedure

Add your preferred method for Don Hall. You will receive an SMS text message, an email to your mailbox, or another of your defined response methods.

Administrative users need two verification options by default.

Log in as `admin@domain.onmicrosoft.com` to `https://myapps.microsoft.com`, and you will see the request for two verification options.

In the next steps, we will verify the functionality.

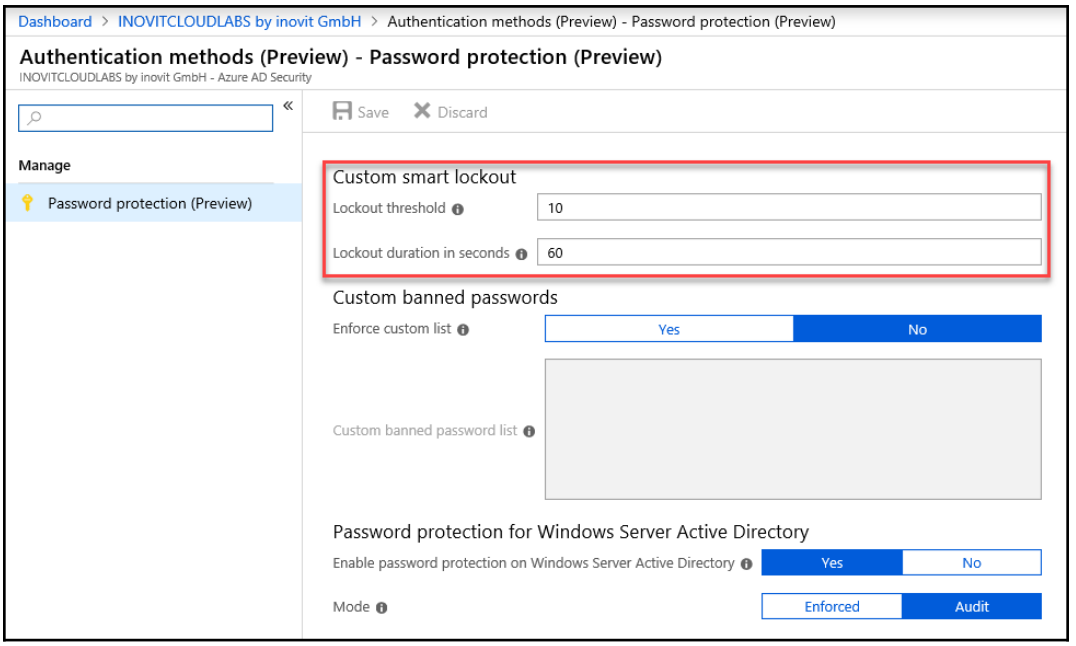
Test the password reset process

Open `https://myapps.microsoft.com` in your preferred browser and enter `Don.Hall@domain.onmicrosoft.com`. Click the **Can't access your account**? option or use the following link, `https://passwordreset.microsoftonline.com`, to start the password reset process. You will come into the verification process, and you need to follow the tasks. Finish the process and log in with the new password.

Using standard security monitoring

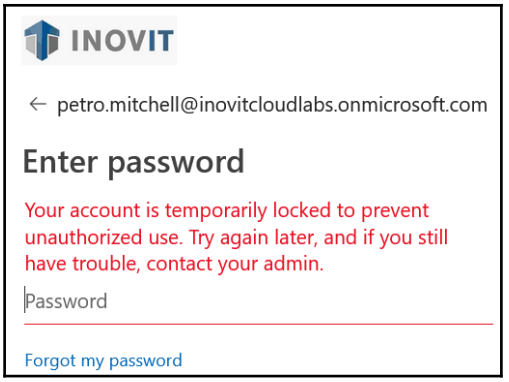
In this section, we will configure and simulate some typical events that get reported in the Azure AD Monitoring section.

First, we configure a **Password protection** feature, **Custom smart lockout**. We set the value to `10` incorrect logins:



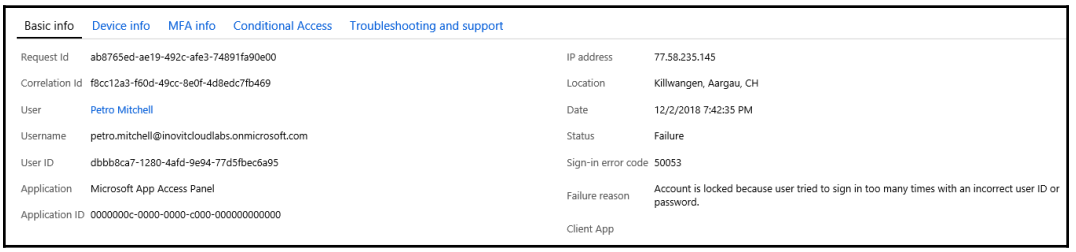
Azure AD password protection features

You should receive the following message if you provide a wrong password 10 times:



Locked message dialog

You can see the activity under **Monitoring** | **Sign-In**:



Azure AD monitoring capabilities

You can also test **Sign-ins from multiple geographies** with simulation software such as **CyberGhost** (`http://www.cyberghostvpn.com/en_us`). Another option would be to use an Azure Virtual Machine.

Log in with an account between geographic regions that are far apart, such as Europe and Asia. This requires a remote machine from your location and in a different time zone, with logons as close together as possible:

- Log in to `https://myapps.microsoft.com` as `Don.Hall@domain.onmicrosoft.com` from your local PC
- Log in to `https://myapps.microsoft.com` as `Don.Hall@domain.onmicrosoft.com` on a machine in a different time zone than your original PC

To configure users with an anomalous sign-in activity, you can use the Tor browser:

- Utilize an anonymous browsing tool such as Tor
- Download the secure Tor browser from `https://www.torproject.org/download/download-easy.html.en`

Open the Tor browser, go to `https://myapps.microsoft.com`, and log in as `Don.Hall@domain.onmicrosoft.com`. Your user account will be locked.

The following result is expected in security monitoring:

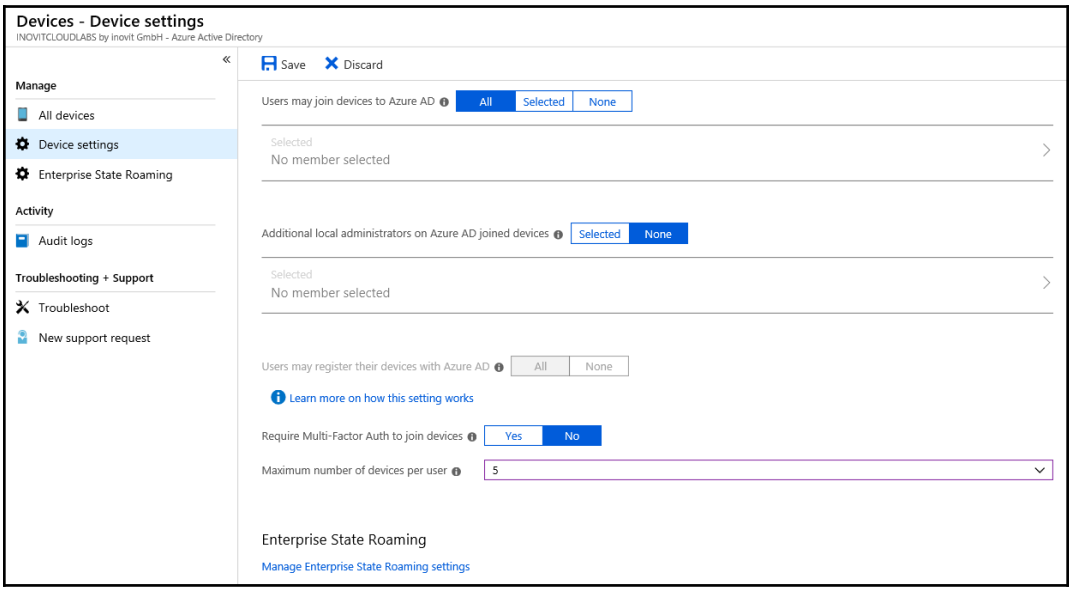
Security monitoring overview - Azure AD

Now that we have had a short journey through the security-monitoring options, we will integrate our Windows 10 client into Azure AD.

Integrating Azure AD Join for Windows 10 clients

In this section, we will configure the Azure AD Join functionality and join our first Windows 10 client to Azure AD.

We configure a maximum of five devices per user and leave the other default values:

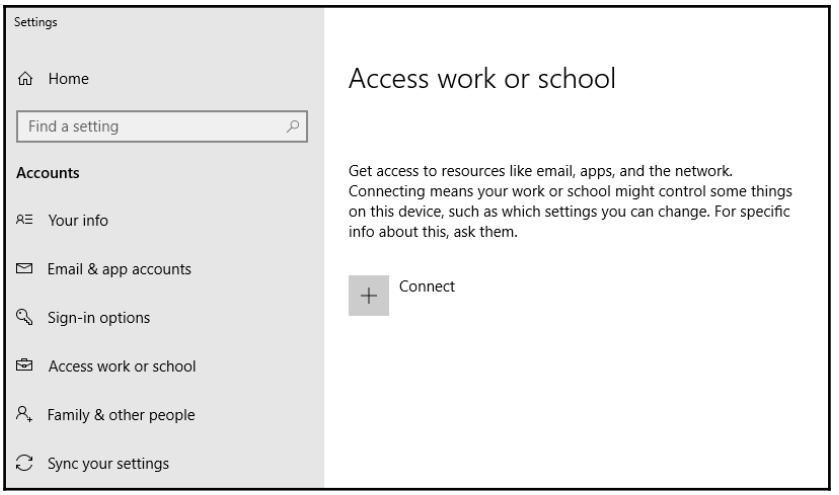


Azure AD - Device settings

In the next section, we will join our client to Azure AD.

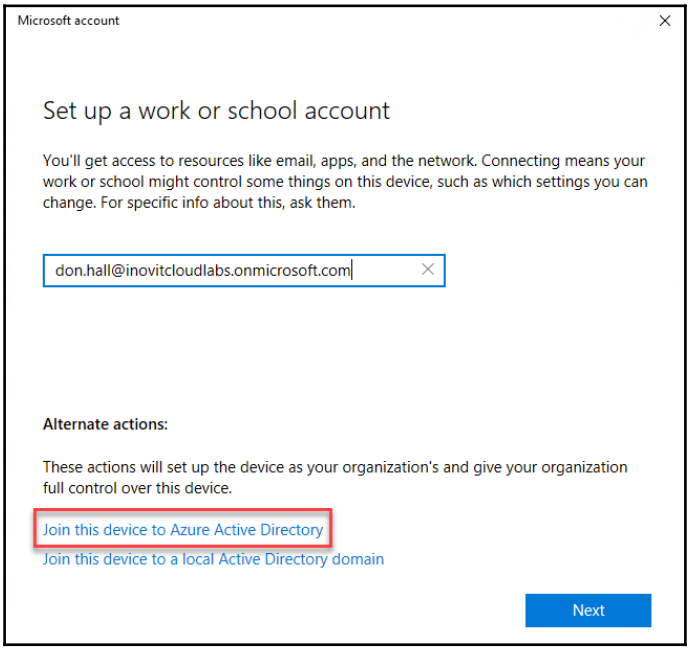
Join your Windows 10 client to Azure AD

Log in to your freshly installed Windows 10 client machine and go to **Settings**. Choose **Connect** in the **Access work or school** section:



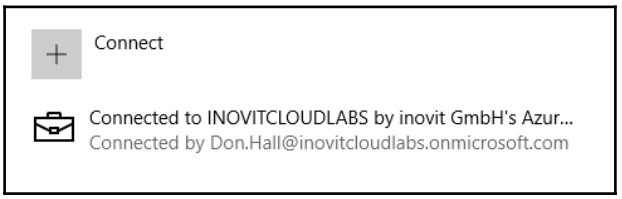
Azure AD Join process dialog

We sign in with `don.hall@domain.onmicrosoft.com` and join the Windows 10 client to Azure AD:



Join actions overview

Click through the **Next** sections and finish joining the client. Afterwards, we will check the new status. The expected result will be the connection to your Azure AD name:



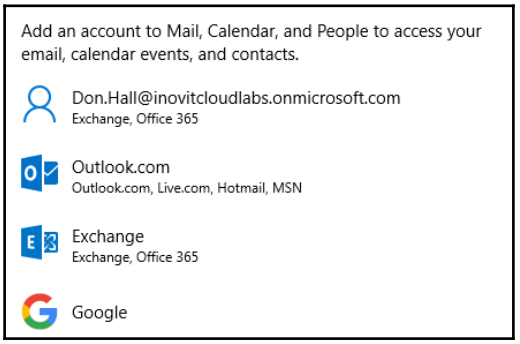
Azure AD joined client message

Afterward, we will verify the Azure AD Join process.

Verify the newly joined Windows 10 client

Log in to the Windows 10 client with the credentials of `don.hall@domain.onmicrosoft.com` and click through the security policy configuration. Click **Enforce these policies**. Click through the PIN setup and finish the process, then test the user experience:

- Open the mail application, and you will see that the system recognizes your user ID and **Single Sign-On** is provided.
- Also, if you open `https://myapps.microsoft.com`, you will be directly logged in to the access panel UI:

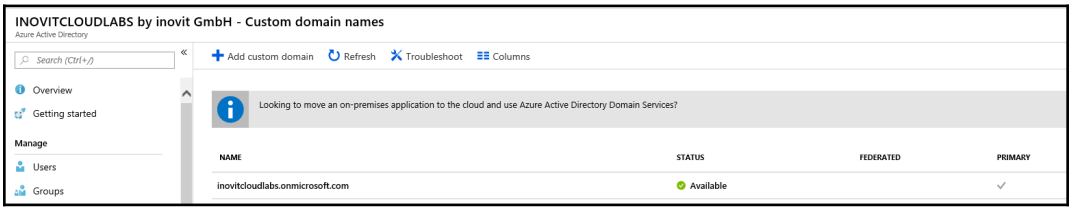


Different mail account options

After verifying the Azure AD Join, we will configure a custom domain. Be aware that you need to register a domain if you want to test the associated functionality.

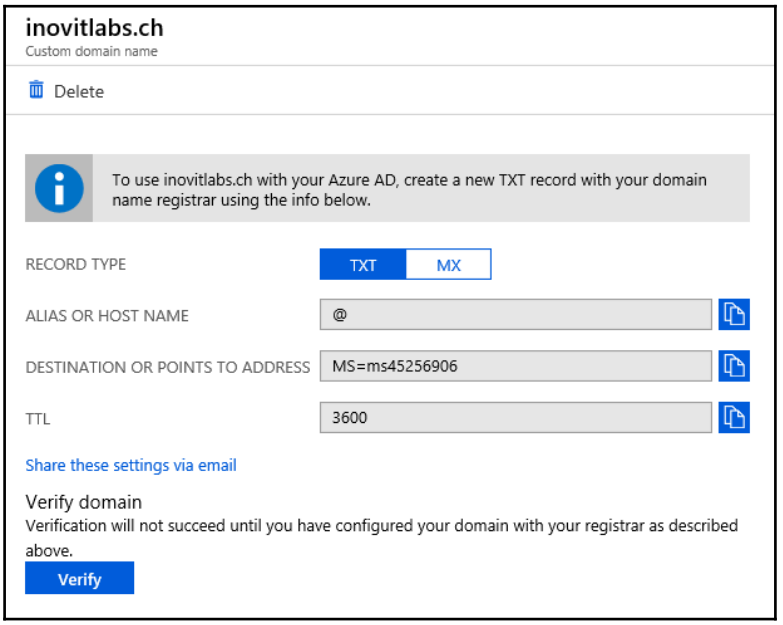
Configuring a custom domain

Under the **Azure Active Directory** | **Custom domain** section, click **Add custom domain** and complete the verification process to prove that you are the owner of the domain:



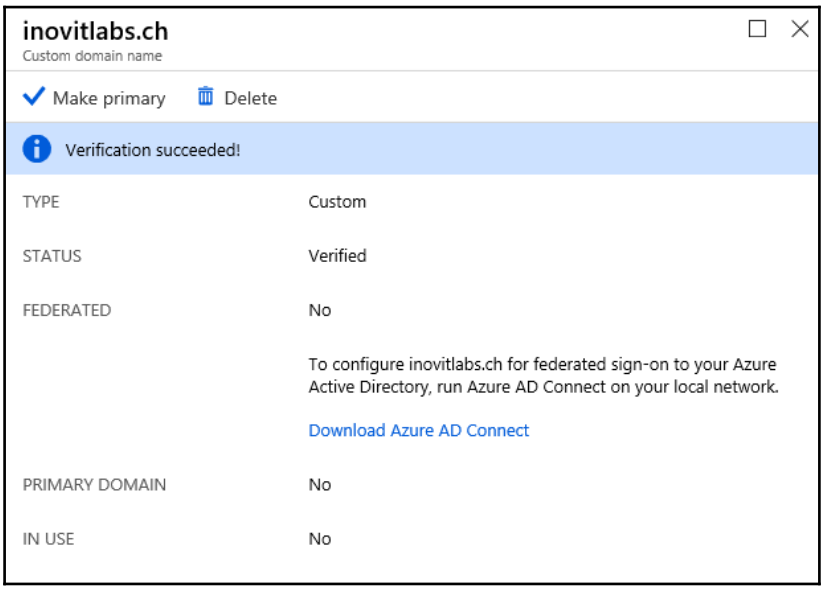
Actual configured domains

Add the **TXT** entry shown to your DNS zone to verify the domain:



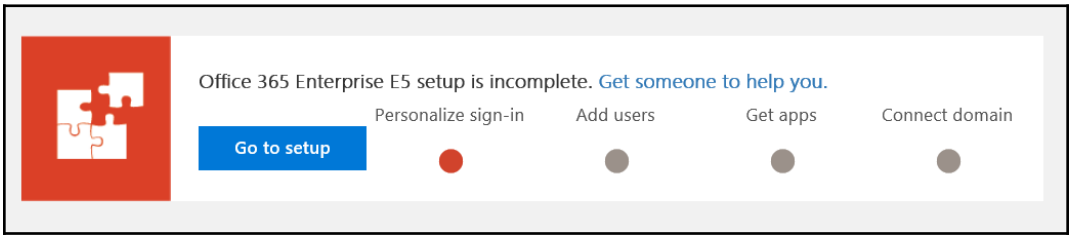
Domain verification options

Click the **Verify** button on your Azure portal, and after successful verification, the new `DOMAIN NAME` will appear under **DOMAINS**. Choose the **Make primary** option:



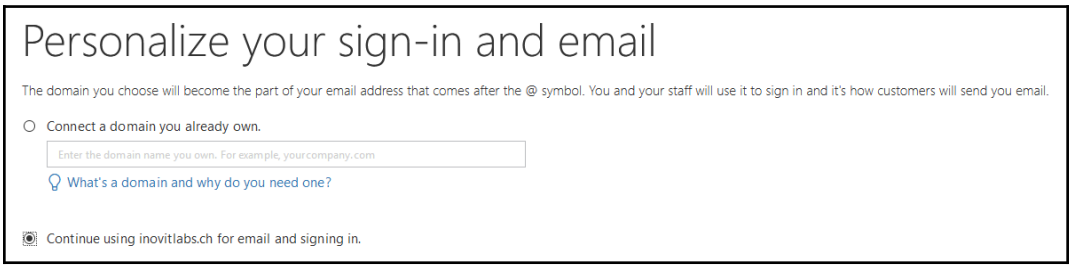
Custom domain overview and configuration options (Make primary or Download the Azure AD Connect tool)

Open `https://portal.office.com` to complete the domain setup process under the admin section:



Office 365 setup wizard

Choose the custom domain to be used for email addresses:



Sign-in and mail options

The last step we need to take is to set the new `UserPrincipalNames` to the existing users. We do this with a small example scripting solution:

1. Connect to your Azure AD with your global administrator credentials:

```
Connect-AzureAD
```

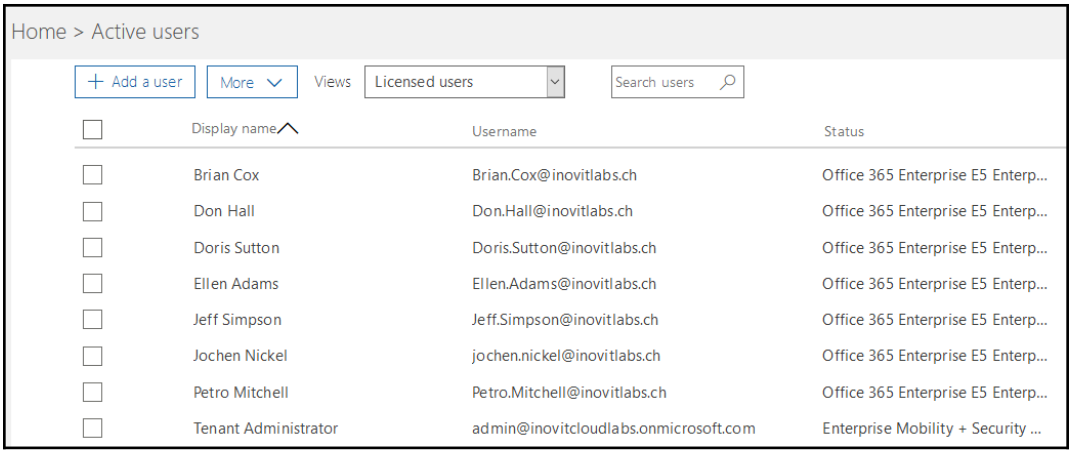
2. Export the existing users to a CSV file with the following cmdlet:

```
Get-AzureADUser -All $True | Where {
$_.UserPrincipalName.ToLower().EndsWith("onmicrosoft.com")} |
Export-Csv C:\Office365Users.csv
```

3. Remove all accounts you don't want to modify and make the change with the following cmdlets:

```
$domain = "inovitlabs.ch"
Import-Csv 'C:\Office365Users.csv' | ForEach-Object {
$newupn = $_.UserPrincipalName.Split("@")[0] + "@" + $domain
 Write-Host "Changing UPN value from: "$_.UserPrincipalName"
to: " $newupn -ForegroundColor Green
 Set-AzureADUser -ObjectId $_.UserPrincipalName -
UserPrincipalName $newupn
}
```

4. You should get a result similar to this:



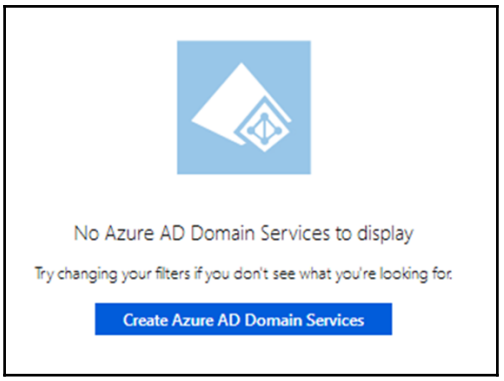
Active users overview

The primary email will also be changed to the custom domain.

Next, we will configure the Azure AD Domain services to provide a transition scenario for a Kerberos-based application that is normally provided in on-premises infrastructure.

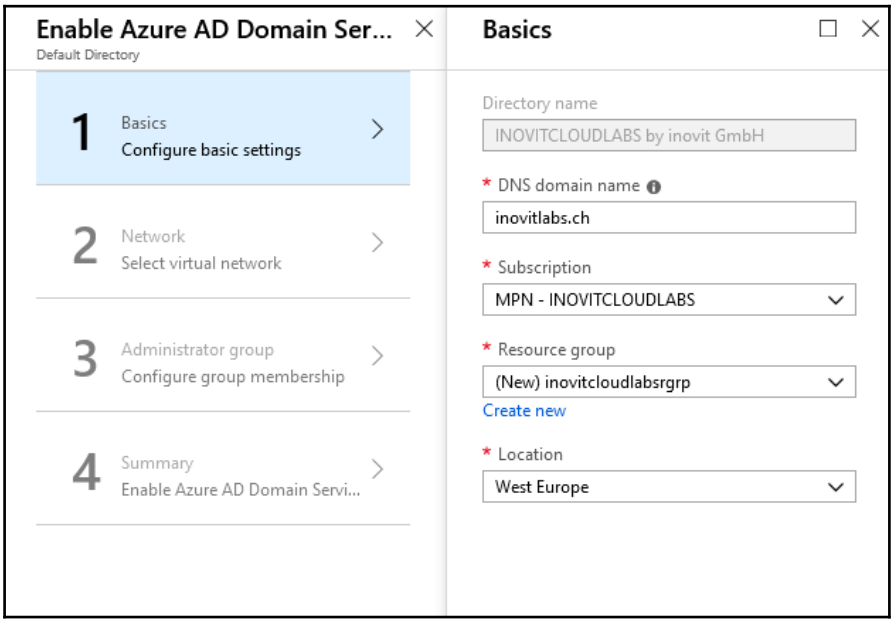
Configure Azure AD Domain Services

To integrate a legacy application based on Kerberos authentication in an Azure **infrastructure as a service** (**IaaS**) scenario, we configure Azure AD Domain Services. In this section, we configure the basic service and integrate an active example application:



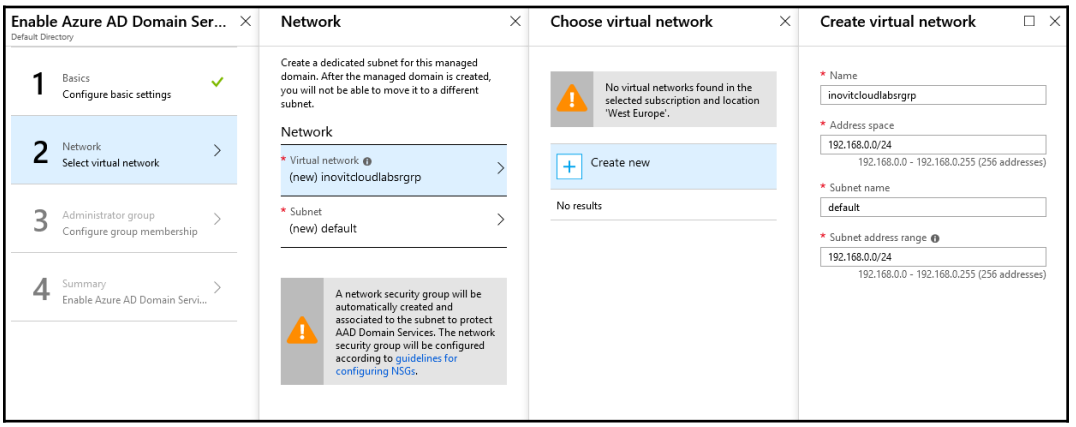
Azure AD Domain Services creation

To start the configuration, we need to specify the **DNS domain name**, the Azure **Subscription** we want to use, and the name of the **Resource group**:



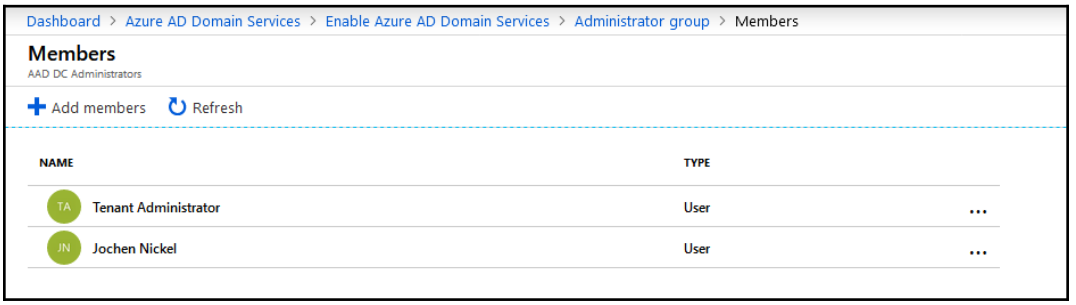
Azure AD Domain Services configuration

When enabling Azure AD Domain Services, you will need to specify which Azure virtual network to use. We use a range `192.168.x.x/20` to configure the network:



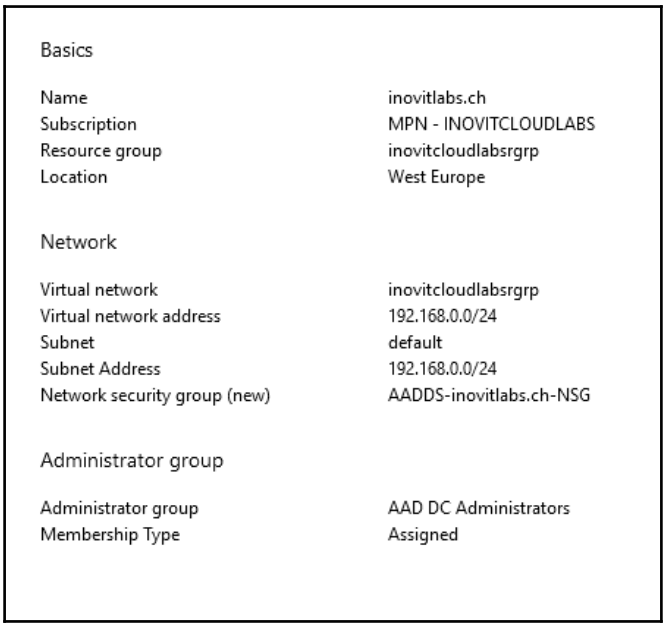
Virtual network configuration

Add the admin account and your test user as a member of the **Azure AD Domain Services Administrator group**:



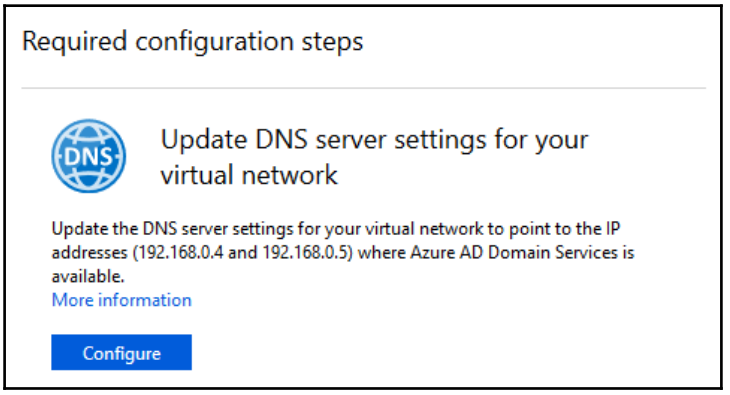
Azure AD Domain Services Administrator group members

The summary should look like the following:



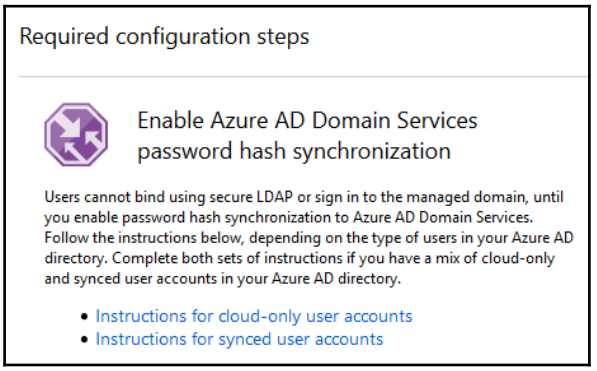
Configuration summary

Next, you will be asked to update the DNS configuration to the addresses of your DNS servers provided by Azure AD Domain Services. In my case, these addresses were `192.168.0.4` and `192.168.0.5`:



DNS configuration

The last important step that you need to complete to use the domain you have just created is to enable password synchronization:



Instructions to synchronize users

By default, Azure AD does not store the credential hashes required for Kerberos authentication. You need to populate these credential hashes in Azure AD so that users can use them to authenticate against the domain. The process can be completed by changing the password of the user. You can use the accounts after 20 minutes in Azure AD Domain Services.

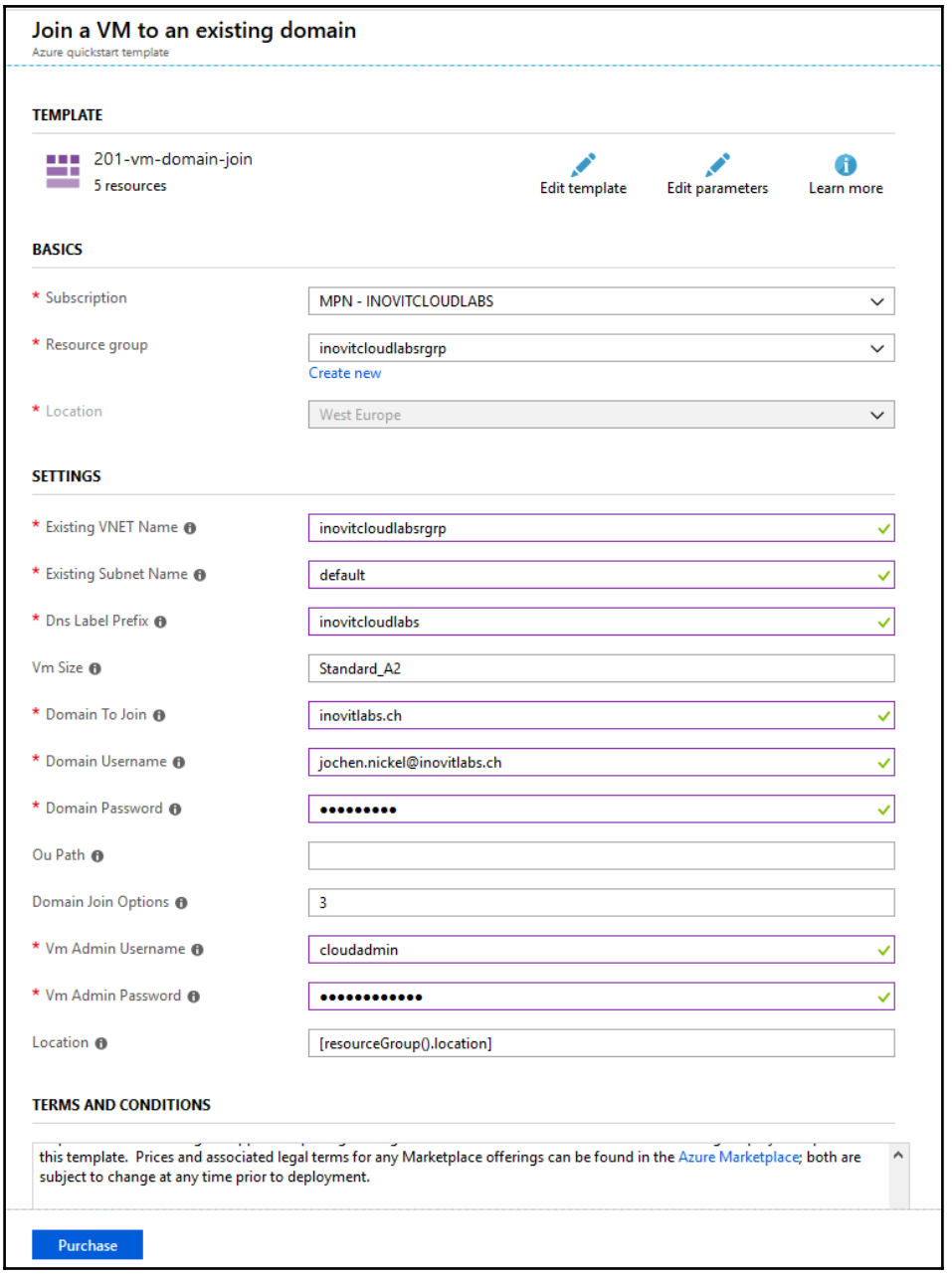
You have two options: let passwords expire for all users or instruct these end users to change their passwords.

Users can use Azure AD's self-service password change mechanism from the Azure AD Access Panel page to change their passwords.

Test and verify your new Azure AD Domain Services

To test the Domain Services, we complete the following tasks:

1. Install a virtual Windows Server in your Azure IaaS environment by using a deployment template (`https://docs.microsoft.com/en-us/azure/active-directory-domain-services/active-directory-ds-join-windows-vm-template`):

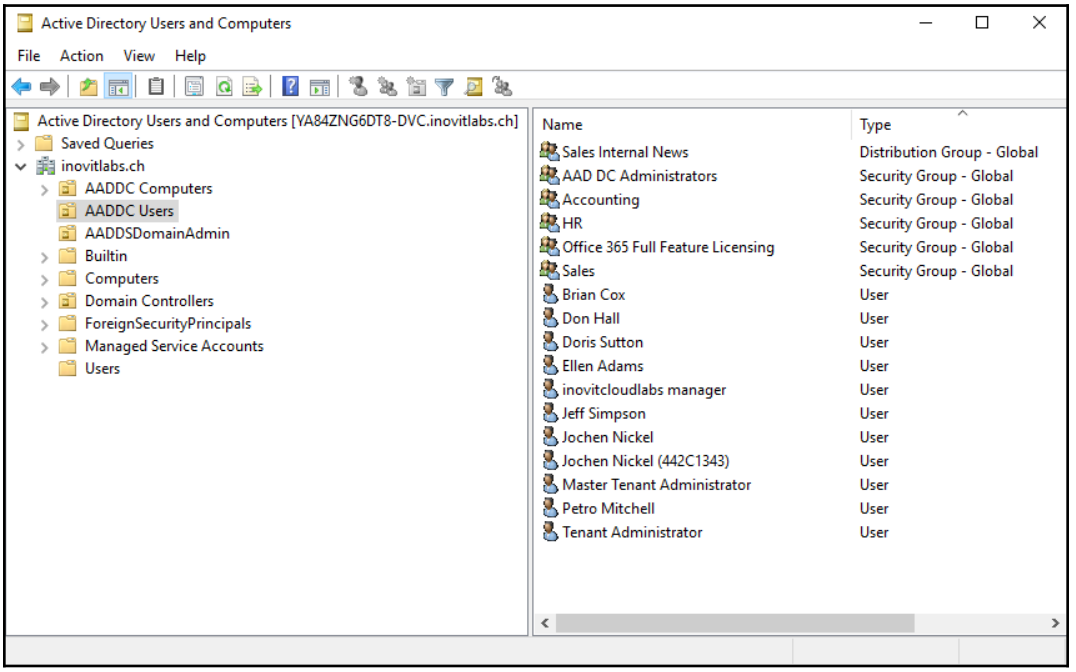


VM deployment configuration

2. Install the administrative tools for Active Directory and DNS on the newly joined server:

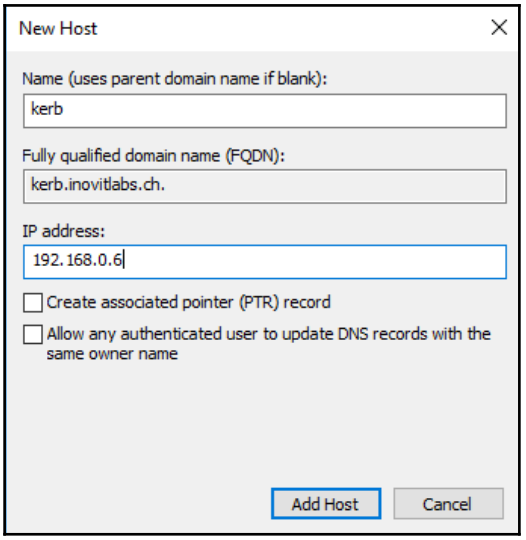
```
Install-WindowsFeature RSAT-ADDS,DNS-Server-Tools
```

3. Connect to **Active Directory Users and Computers** (`dsa.msc`) and the **Group Policy Management** console to verify your configuration:

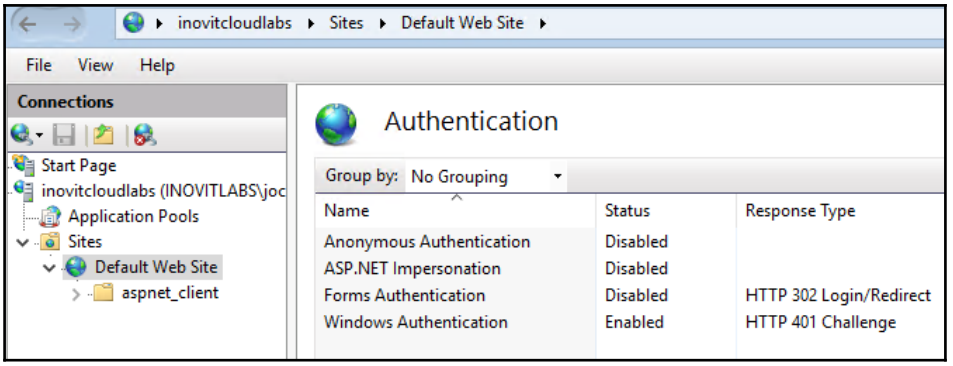


Azure AD Domain Services structure including synchronized objects

4. Next, we need to create a **DNS HOST (A)** record for our test application:



5. Now, we can install a basic IIS configuration, used to handle the Kerberos part. For this, you need to install the IIS components, choose the Kerberos authentication feature, and activate it on the default website. Only **Windows Authentication** needs to be activated:

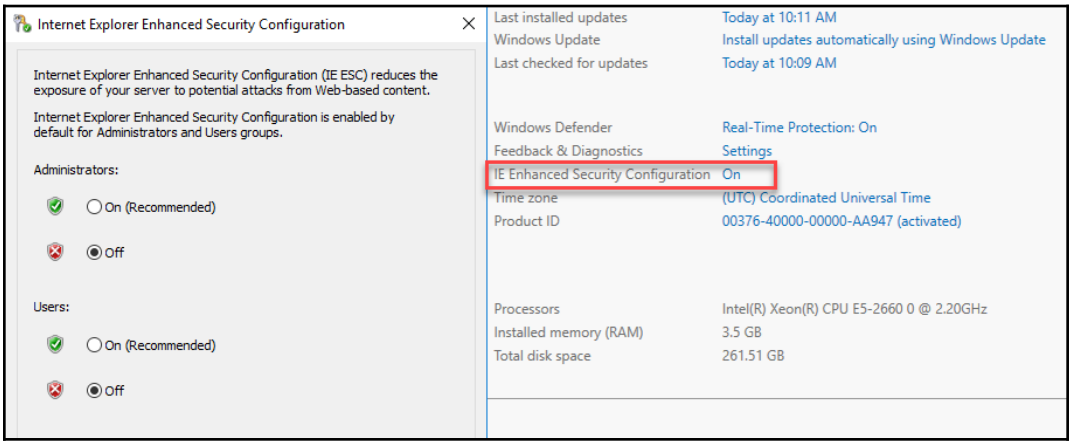


IIS Authentication configuration for Kerberos example application

6. Next, we will install and configure the Azure AD App Proxy connector to provide the application to your users. We use the following cmdlets to configure the needed, resource-based KCD feature:

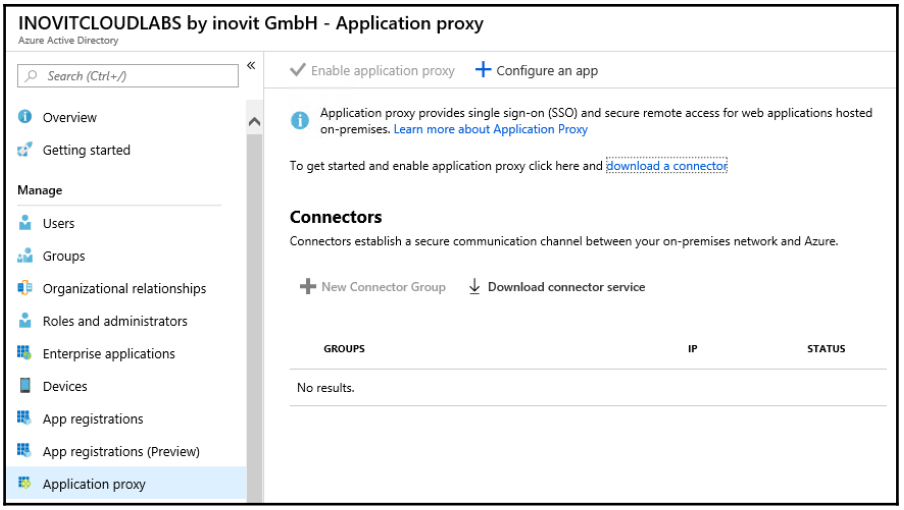
```
# inovitcloudlabs represents the computer name
$ConnectorComputerAccount = Get-ADComputer -Identity
inovitcloudlabs
Set-ADComputer inovitcloudlabs -
PrincipalsAllowedToDelegateToAccount $ConnectorComputerAccount
setspn -S HTTP/kerb.inovitlabs.ch inovitlabs\inovitcloudlabs
```

7. Next, we will activate and configure the Azure AD App Proxy. To make it simple, we disable the **IE Enhanced Security Configuration** so that we don't need to provide any IE Security Zone configurations, just for the lab:



Server Manager IE Enhanced Security Configuration

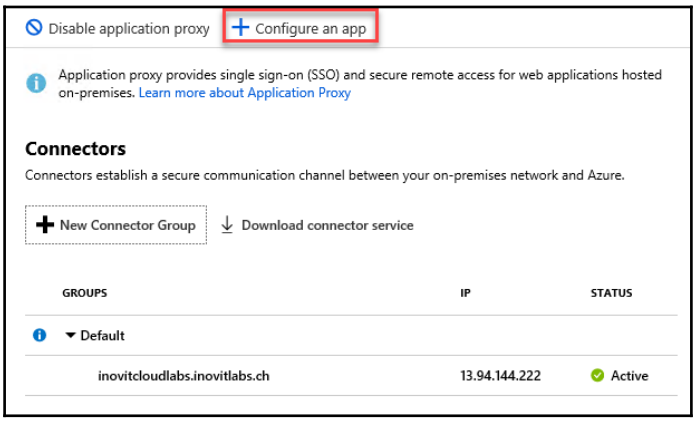
8. Next, we need to download the connector and install it on the server:



Application Proxy agent download and configuration

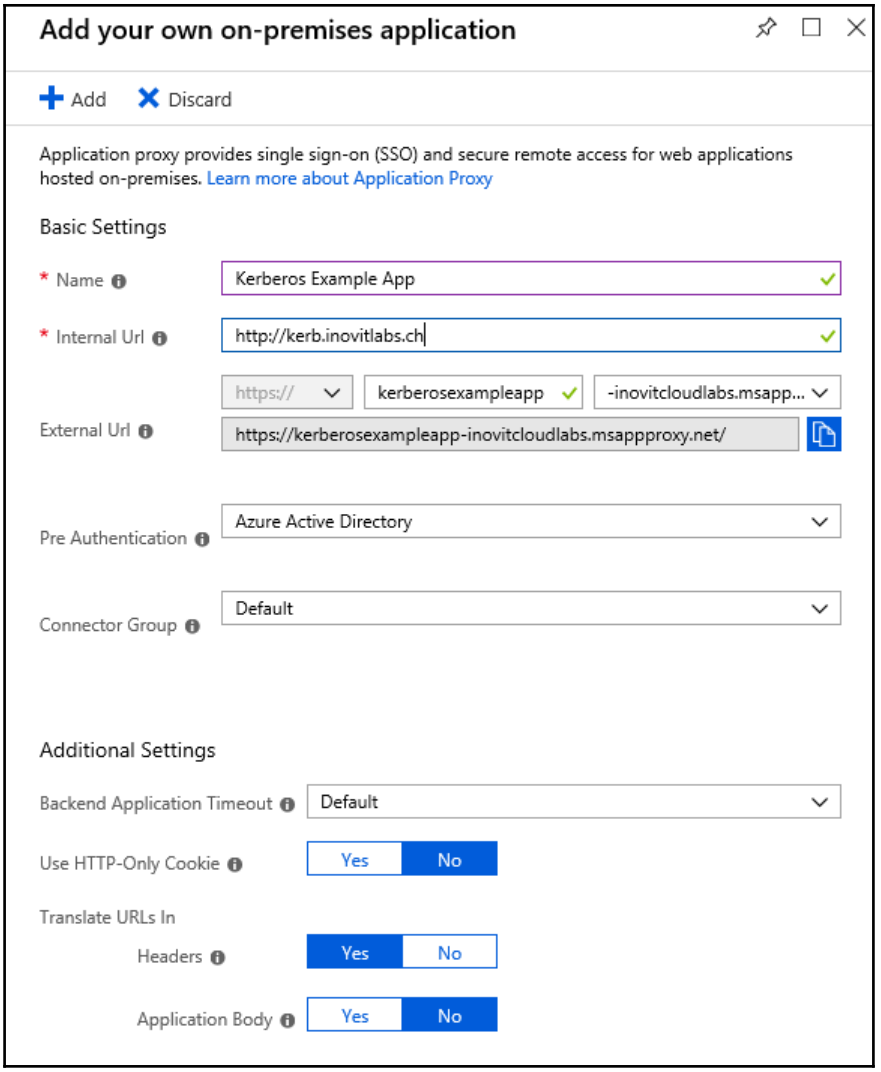
To configure the connector on the server, you need to provide a user with global administrator rights.

9. After installing and configuring the connector, we will add our example app:



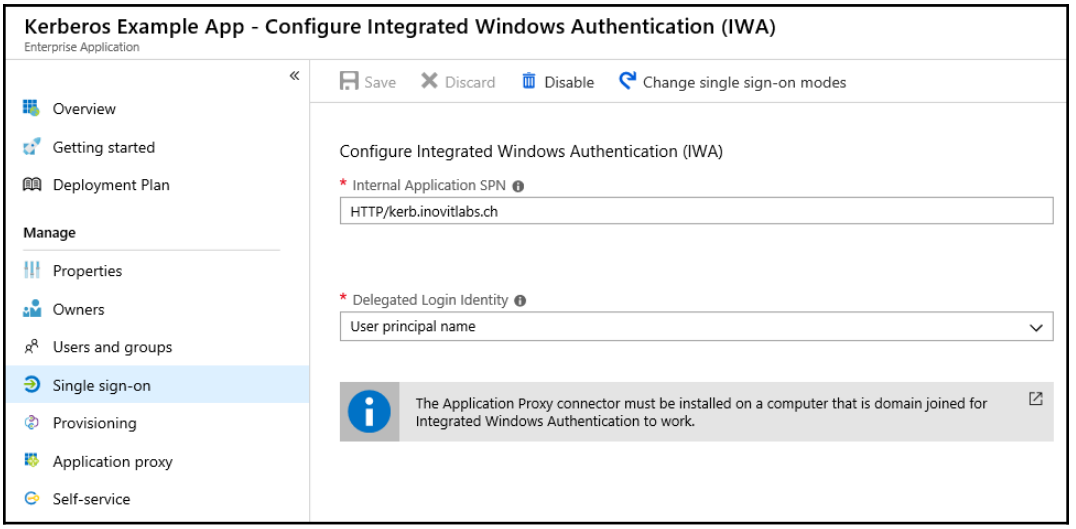
Azure AD App Proxy Connector group configuration options

10. Next, we configure our example app as shown:



Kerberos example configuration

11. Next, we configure the **Integrated Windows Authentication** (**IWA**) option:



Application IWA configuration

Finally, we assign some users or groups and test the application at `https://myapps.microsoft.com`. As a result, you should see the IIS test page. We provided a sample Kerberos-based application to Azure AD Domain Services and used the Azure AD App Proxy functionality.

Summary

After working through this implementation scenario, you will be able to configure and manage a suitable Azure AD tenant for the most important tasks. You will also be able to integrate Windows 10 and Office 365 to build a productive workforce for your users without an on-premises infrastructure. Don't worry if you missed some functionality. This was just a warm-up.

In the next chapter, we will discuss the identity synchronization needed to start with your hybrid integration and to provide the correct identity synchronization scenario for your requirements.

2
Understanding Identity Synchronization

The main component in a hybrid identity and access management solution is the connectivity between the on-premises **Active Directory** (**AD**)and the **Azure Active Directory** (**AAD**), including the related synchronization of objects and attributes. Microsoft tries to make the synchronization process straightforward without administrators needing to have the complete details of the system under the hood.

In this chapter, we'll discuss the essential identity-synchronization scenarios and tools for the successful implementation of a full hybrid identity life cycle management. We'll start with an overview of the **Microsoft Identity Manager** (**MIM**) and the Azure AD Connect tool, and then we can dive into the identity-synchronization scenarios. Afterward, we'll run through the different processes, the AD user account cleanup for a hybrid environment and all the crucial parts and steps of the identity synchronization in Azure AD Connect. The chapter will be rounded up with a lot of practical tips and use cases.

We'll cover the following essential topics:

- Technology overview
- Synchronization scenarios
- Synchronization terms and processes

In the first section, we start with the technology overview.

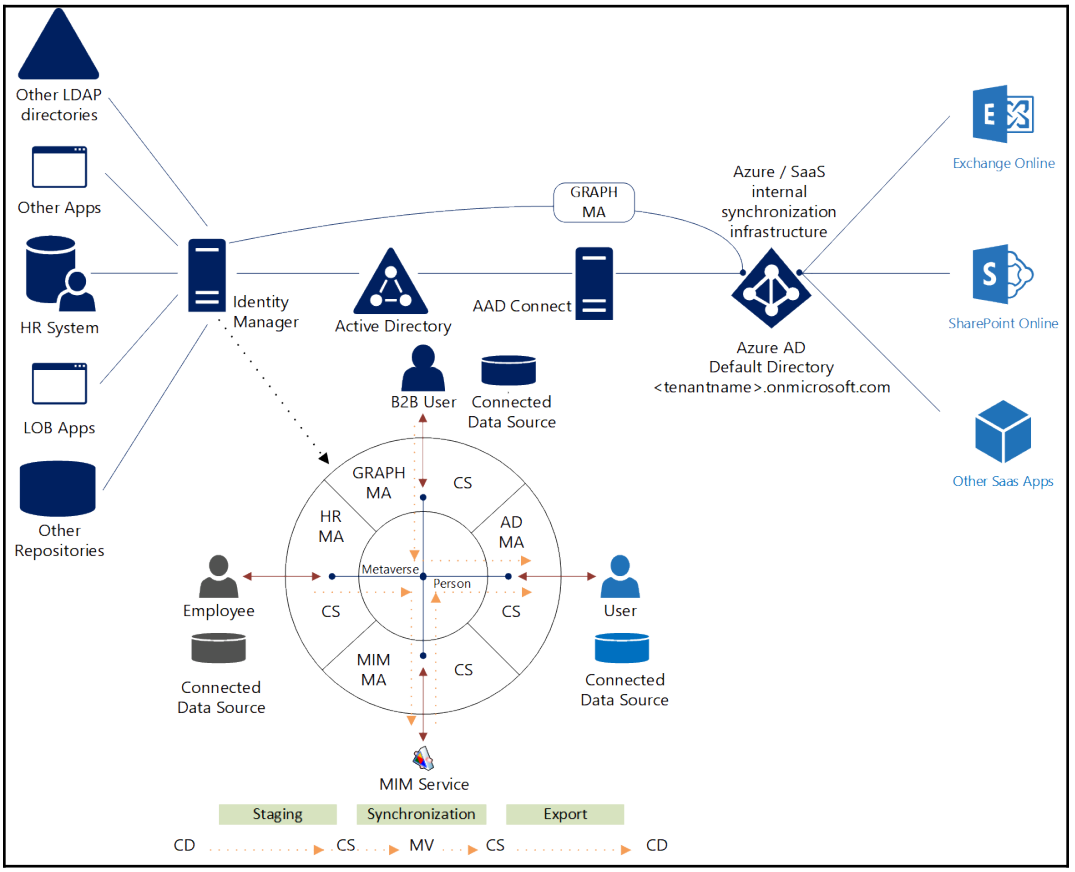
Technology overview

The **Microsoft Identity Manager** (**MIM**) 2016 or other identity management products are typically used to prepare the identities stored in the local Active Directory for cloud synchronization. The Azure AD Connect tool is generally used to synchronize the AD identities to the Azure AD to be used in connected **software as a a service** (**SaaS**) applications or other functionalities. The main advantage that MIM 2016 provides for this solution is to help with domain/forest consolidations, attribute normalization, and complete on-premise identity management with the help of workflows to support your business processes.

As you can see in the following diagram, MIM 2016 is also capable of synchronizing identities with the Azure AD. So, you're probably wondering which tool you should use to sync identities with Azure AD.

The short, practical answer for common scenarios is the Azure AD Connect tool because it supports all the provided synchronization functionality of the Microsoft Azure AD.

The following diagram provides a schematic view of the usage scenarios of both tools:

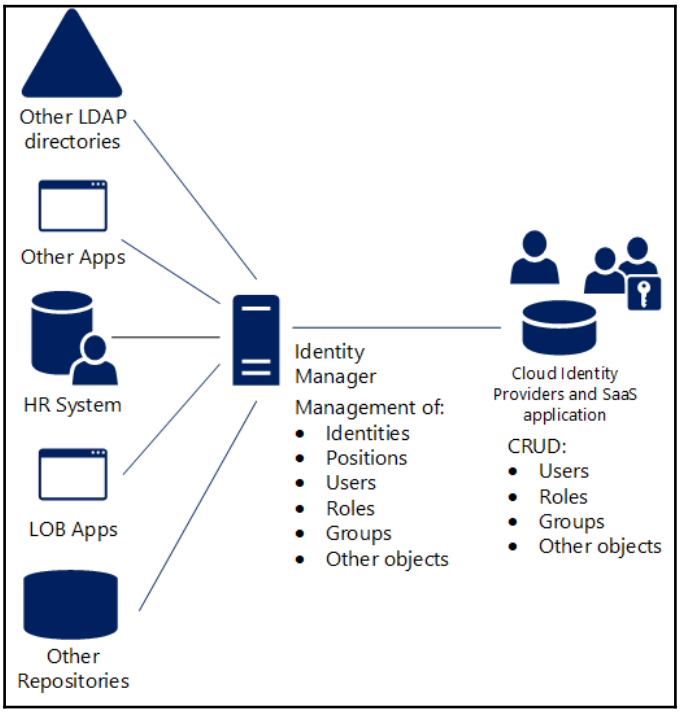


Identity synchronization architecture

Azure AD Connect doesn't offer an active user write back. You'll find this option is deactivated in the Azure AD Connect configuration. To add this functionality, you can use the MIM Graph API connector like in Azure B2B user management, where you need to write the guest user back to your AD. To view a comparison between the tools, check out *Hybrid Identity directory integration tools comparison* at `https://docs.microsoft.com/en-us/azure/active-directory/hybrid/plan-hybrid-identity-design-considerations-tools-comparison`.

Microsoft Identity Manager (MIM) 2016

MIM 2016 is the primary identity and access management product of Microsoft that provides all the different server roles and components needed in this field of technology. MIM 2016 is mainly used to provide a sanitized and central identity in on-premise environments. In the context of a hybrid architecture, it plays a crucial role in connecting any repository to manage identities in different repositories. Furthermore, complex identity-management scenarios are provided with this component. This also includes the management of Azure AD and many SaaS applications in today's market, as you can see in the following diagram:



Identity Manager functionality and objects

The following section gives you a short overview of the key components of MIM 2016 to help a solution architect/engineer to identify possible interactions or elements that need to be included in a design blueprint for a suitable solution. We also use some of these components in the provided implementation guides of the book, such as in `Chapter 8`, *Using the Azure AD App Proxy and the Web Application Proxy*, where we implement the Azure AD **business to business** (**B2B**) scenarios.

The following main feature sets are provided by MIM 2016:

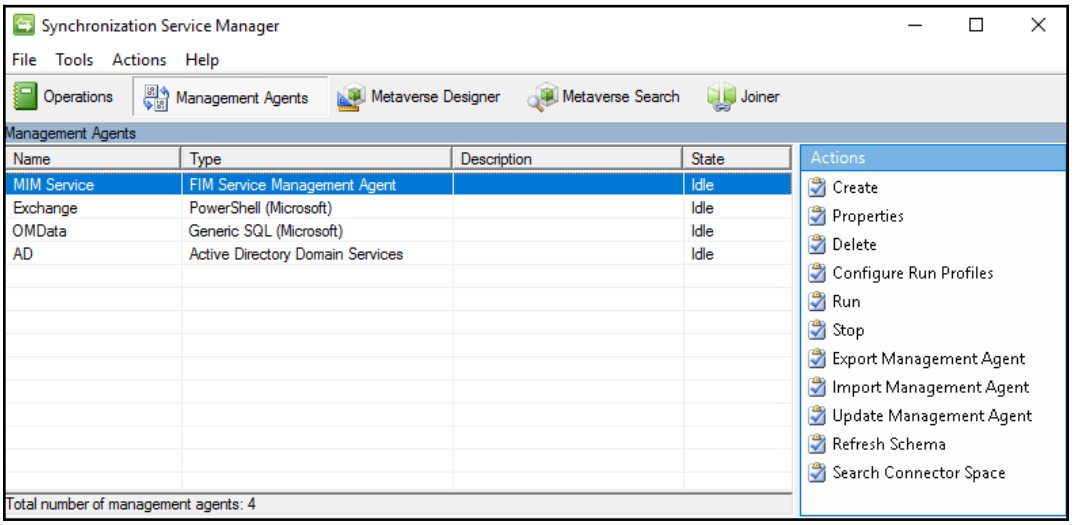
- Identity synchronization including provisioning/deprovisioning
- Access request and Access Policy Management
- Delegation of administration
- Self-service password reset and account unlock
- Self-service group management
- Role management (RBAC, ABAC, SoD)
- Manual managed groups
- Manager-based groups
- Criteria-based groups (attribute-based access controls)
- Time-limited group memberships
- Certificate management
- Reporting and compliance and Access Certification

If you want to use MIM 2016 as your central identity-management system, we highly recommend you take a look at the **Workflow Activity Library** (**WAL**) under `http://microsoft.github.io/MIMWAL/`. Also, the combination of the newly integrated privileged access-management solution in Windows Server 2016 and MIM 2016 provides a very effective way to manage and limit security issues with administrative accounts.

MIM synchronization service

The MIM synchronization service is the heart of the product. You can import and aggregate data in a central identity store, which is called **metaverse** (**MV**). The service provides a staging zone to compare the connected directory with the metaverse. This staging zone is called the **connector space** (**CS**). The connection to any repository is done over specific connectors, which are also called **management agents** (**MAs**). The synchronization service is also responsible for the data integrity through all connected repositories, including the provisioning and deprovisioning of objects, including their attributes. You can find an overview of the available connectors at `https://docs.microsoft.com/en-us/microsoft-identity-manager/supported-management-agents`. In case you need a custom connector, MIM provides a development framework to fit your needs.

The following screenshot shows an example of different directories and systems that can be connected:



Microsoft Identity Manager Synchronization Service Manager

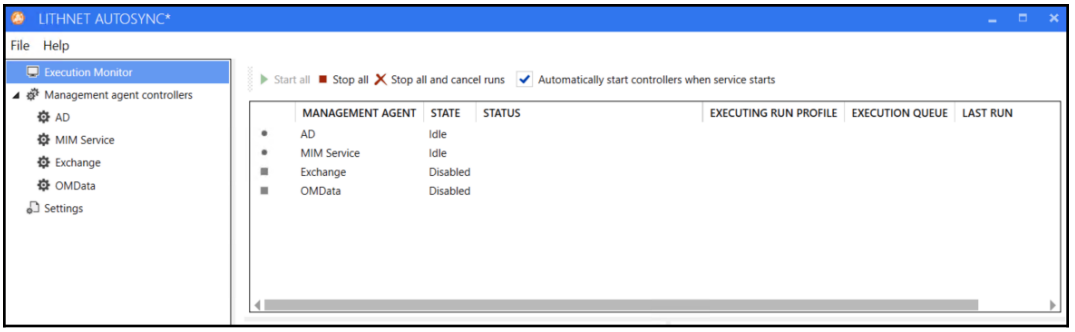
Refer to the following practical notes:

- MIM synchronization service is a state-based system by default:
    - Imported data will be compared with previously imported data; differences indicate modifications in the source
    - Imported data will be compared with previously exported data; no difference indicates a successful export
- MIM is not real-time; it uses run cycles that can be configured

MIM synchronization can work event-based with a synchronization service extension, as we provide in the following section.

MIM synchronization service extensions

To extend the functionality of the MIM synchronization service, you could use the **Lithnet AutoSync. Lithnet AutoSync** runs as a Windows service in addition to the MIM synchronization service. The service enables you to execute MIM Management Agent run profiles automatically. Furthermore, **Lithnet AutoSync** empowers you to perform exports and delta synchronizations automatically if the MIM synchronization engine detects a change. Now you have an event-based and a run-cycle-enabled synchronization engine. The following screenshot shows the configuration on the sync engine that you saw in the preceding screenshot:



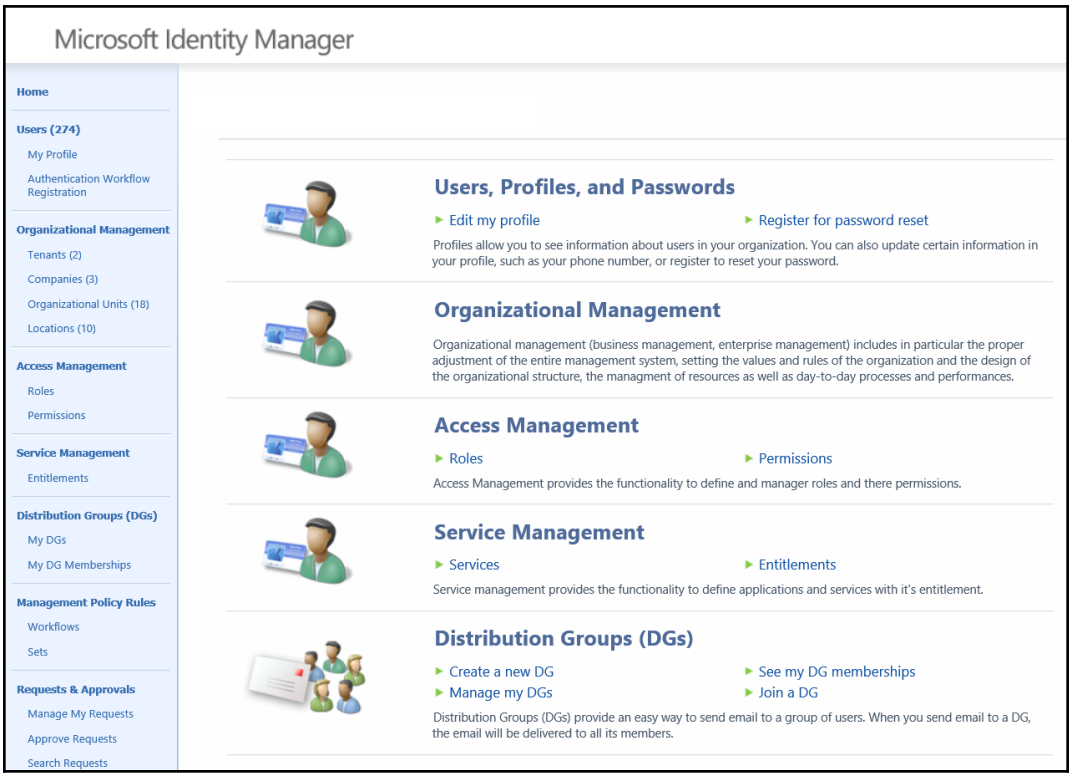
Lithnet Autosync management console

To learn more about the **Lithnet AutoSync** tool, check out `https://github.com/lithnet/miis-autosync`.

MIM service and portal

The MIM service provides all the necessary capabilities for policies and workflows. All objects like groups, users, workflows, requests, and other resources used in MIM are stored as objects in the MIM Service database. These objects can be modified with typical CRUD (**Create**, **Read**, **Update**, **Delete**) operations requests to the MIM service **Identity-Management** (**IdM**) Platform.

With the MIM portal, users interact with the system by using a web browser. The usage depends on the permissions; users can be granted to do requests, respond to approval requests, cancel existing pending requests, or manage objects in the IdM system:



Microsoft Identity Manager Standard Portal

The default MIM Portal is based on a SharePoint solution and can be highly customized. You can find some example configurations at `https://docs.microsoft.com/en-us/microsoft-identity-manager/reference/mim-portal-customizations`.

MIM service extensions

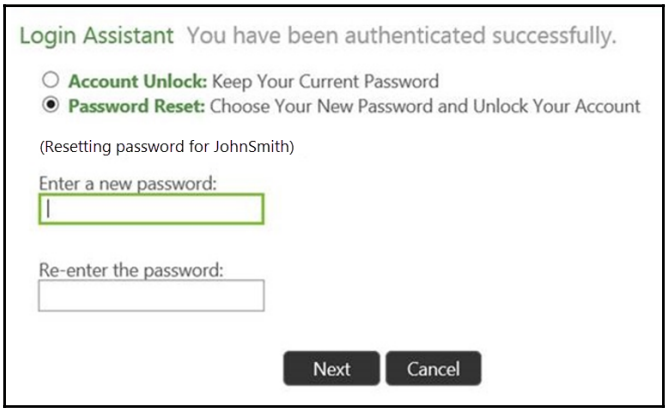
The Lithnet **Forefront Identity Manager** (**FIM**)/MIM Service REST API is a wrapper for the FIM/MIM Service's **Simple Object Access Protocol** (**SOAP**)/**Windows Communication Foundation** (**WCF**) endpoint, exposing create, update, delete, and search functionalities via a series of standard HTTP calls. The API returns JSON-formatted data, making it compatible with a wide range of platforms and services. By default, MIM doesn't provide a REST API to the MIM Service. With this extension, you can adopt your own functionality to MIM 2016, such as your own portal as we've done, or to interact with Power BI and other technologies to provide custom solutions. For more information about the Lithnet REST API for FIM/MIM Service, visit `https://github.com/lithnet/resourcemanagement-webservice`.

MIM password reset and user account unlock

MIM provides two password-related features that can help you to offer solutions in your on-premises environment:

- **Password synchronization**: Password synchronization to other repositories based on the AD password change or reset
- **Password and account self-service**: Separate portals to provide a self-service password reset and account-unlock capabilities

The following screenshot shows the web-based **Password Reset** and **Account Unlock** functionality:



Microsoft Identity Manager Self-Service Password Reset dialog

In particular, if you have older Windows clients, such as Windows 7 or Windows 8/8.1, in your environment, you can provide the **Password Reset** functionality in the Windows login UI. The **Password Reset** functionality in Azure only provides support for Windows 10 clients but delivers more capabilities in the verification options than the MIM solution.

Be aware that MIM 2016 isn't able to provide a password hash synchronization such as Azure AD Connect in a hybrid scenario.

MIM privileged access management

MIM 2016 provides a **privileged access management** (**PAM**) solution, restricts privileged access within an existing AD environment.

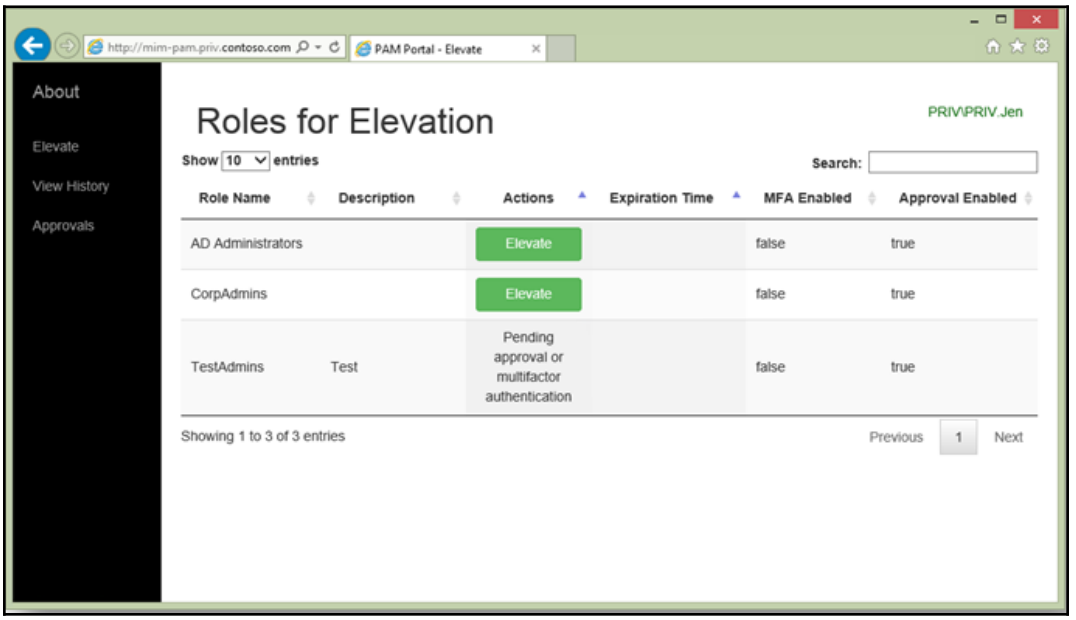
PAM solves the following two targets:

- You can get back the authority over a compromised AD environment if you provide a separate bastion environment that is more protected from malicious attacks
- With the isolation of privileged accounts, you can limit the risk of losing sensible credentials

PAM helps to address the following problems:

- Pass-the-hash and pass-the-ticket attacks
- Kerberos compromises or spear phishing
- Unauthorized privilege escalations
- Other vulnerabilities and attacks

The following screenshot shows you the role-activation and user-verification processes on the MIM PAM example portal, which you can customize based on your needs:



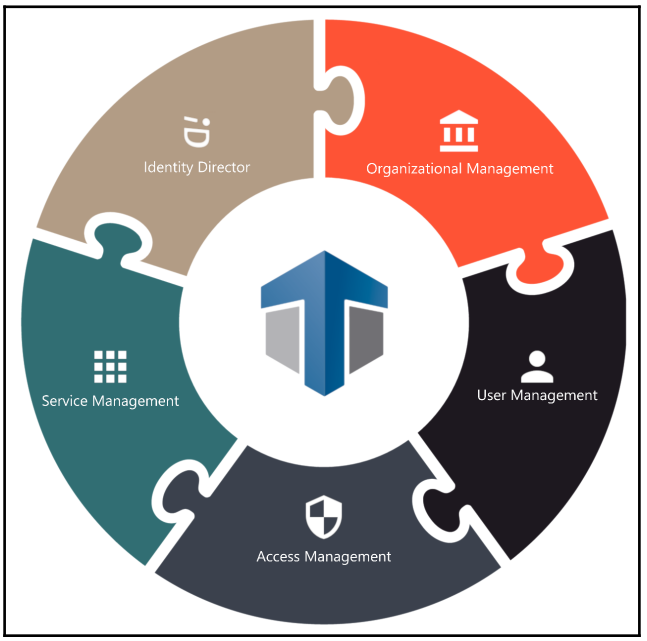
MIM privileged access management sample portal

Now that you know a bit about MIM's standard functionality, we'll provide you with an overview of an additional solution we developed with a partner company. Hopefully, it gives you an idea of the possibilities that MIM provides and how they can be expanded.

Additional solution

Based on our many years of experience and successfully implemented projects in the area of Identity and access management, we've decided to map the recurring requirements in our solution that fill the gaps we couldn't fulfill with the MIM standard functionality.

The following five key pillars will be provided by our solution and enable us to implement highly standardized identity and access-management processes that are flexible and customizable:



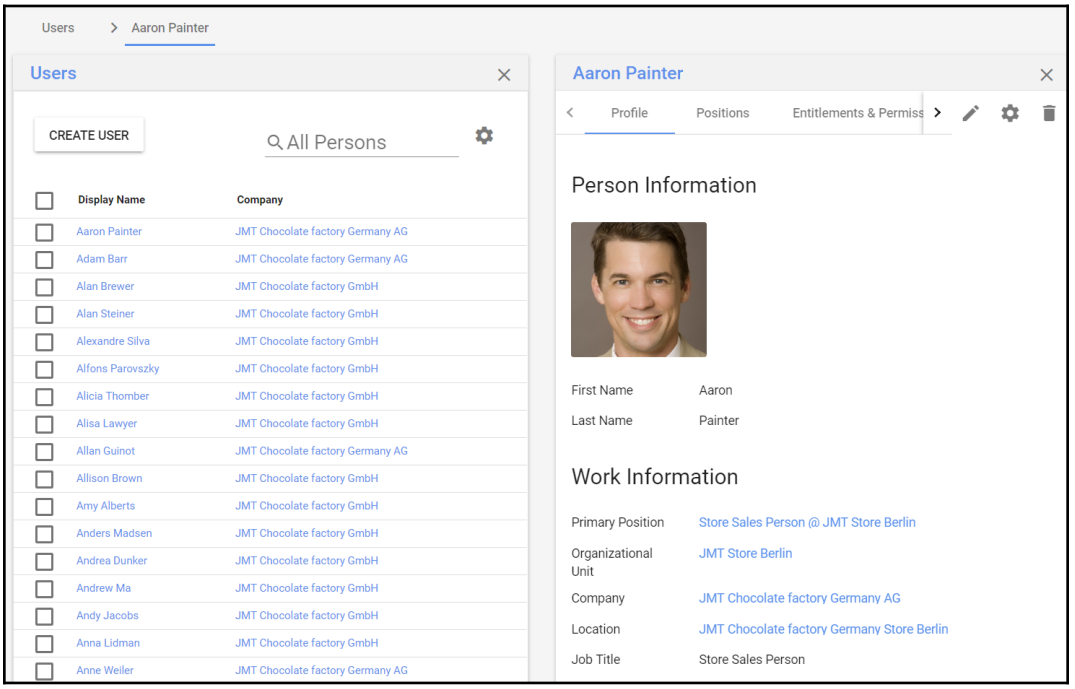
Inovit Identity solution - building blocks

The main features of **Organizational Management** and the frontend are the following:

- Supports several cloud scenarios and traditional IT infrastructures
- Representation of the organizational structure (parameterization and inheritance)
- Management of the organizational structure (manual or synchronized)
- Allows you to build efficient, role-based access controls
- Enables you to deliver beneficial cost management
- User-friendly and highly responsive frontend
- **Single page application** (**SPA**) architecture
- Integrated governance features
- No SharePoint installation required

- Highly customizable
- Single frontend for on-premises or cloud-only deployment
- A clear strategy for future invests
- Cloud management

The following screenshot schematically shows the frontend:

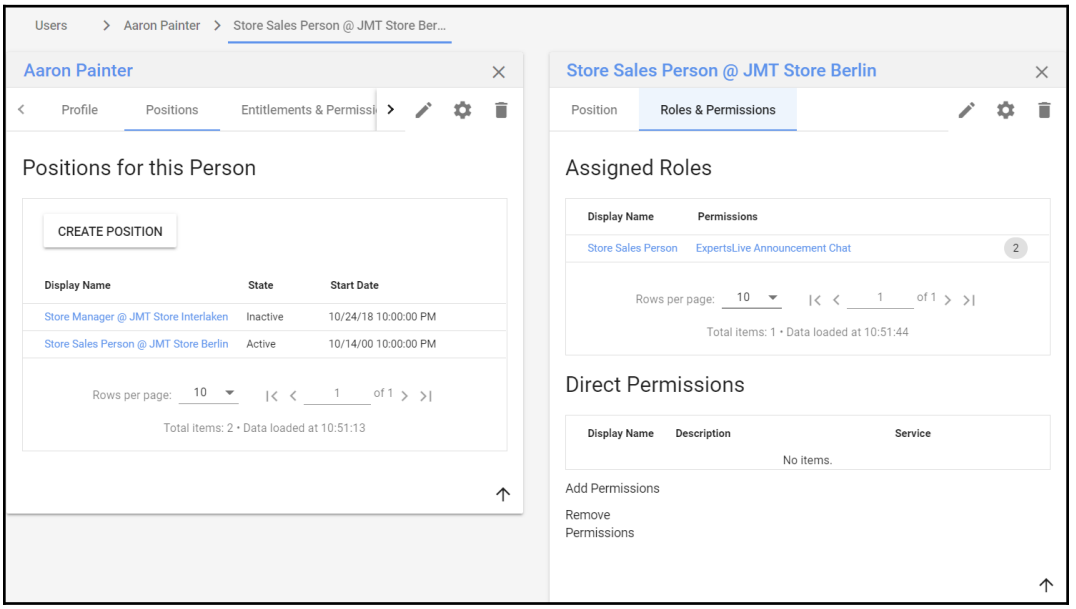


Identity Directory SPA portal

The main features of **User Management** are as follows:

- Standard processes (onboarding, mutation, offboarding)
- Time-limited user accounts
- Management of standard and administrative user accounts
- Management of Azure B2B accounts
- Automatic `samAccountName` and user generation
- Alignment of UPN, email, and SIP for cloud usage
- Password reset and account unlock

The following screenshot schematically shows the positions and role assignment in the frontend:



Identity Director position-based Access Management

The main features of **Access Management** are the following:

- Position-based, role-based, attribute-based access management
- Permissions directly assigned to a user (if required)
- Approval and notification workflow support
- Administration of privileged accounts
- Authorization direct views and reports
- Bidirectional interfacing with services such as SharePoint and Microsoft teams

The main features of **Service Management** are as follows:

- Automated and straightforward adaptation of systems and services
- Representation of the service catalog
- Synchronization-based order units
- Workflow-based order units (notification and approvals)
- Management of Office 365 and other cloud services

Cloud deployment based on identity director service

With our Azure-based next-gen SaaS integration, you'll be able to choose where you want to start your implementation, whether on-premises based on MIM or directly in the cloud. There will also be a fully supported transition path to the cloud. On Azure, we use the Identity Director Service with a Cosmos DB backend that provides the same functionality and additional features that you already have with an on-premises MIM platform. The highly responsive frontend won't change for the user.

On-premises deployment based on MIM 2016

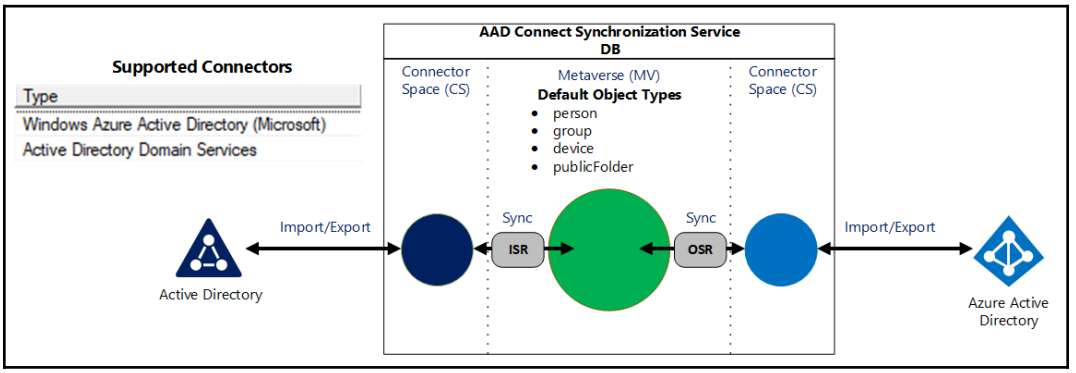
MIM 2016 represents Microsoft's on-premises Identity and Access Management solution. MIM 2016 builds the base platform to our on-premises deployment method. With a robust synchronization framework and many interfaces, such as REST, SOAP, LDAP, SQL, PowerShell and file-based, you can harmonize AD identities for synchronization with Azure AD. The MIM 2016 Service provides a dominant workflow component that works state and event-based to solve the Identity and Access Management requirements of your business. With MIM 2016 in place, you benefit from cloud-ready identities, powerful user self-service, and enhanced security.

Now that we've had a quick look at MIM, and the additional available extensions and solutions for preparing and managing the AD objects, we'll proceed with our connection to the Azure AD.

Azure Active Directory Connect

Azure AD Connect is part of the identity bridge to build a hybrid cloud identity and access management. The main purpose of this tool is to synchronize on-premises identities to the cloud and is limited back to the AD. Azure AD Connect is also able to install and configure the whole identity bridge. This means that you can also install and configure your **Active Directory Federation Services** (**ADFS**) infrastructure for federation reasons with this tool or use pass-through authentication combined with seamless **single sign-on** (**SSO**) to provide a modern and comfortable authentication scenario. The tools are also capable of integrating external federation tools, such as PingFederate.

The following screenshot provides you with a schematic overview of the Azure AD Connect components, whose practical usage we'll explain during the chapter:



Azure AD Connect synchronization schema overview including management capabilities

The following terms are used in the preceding figure:

- **Connected Data Source (CD)**: A data source that can be represented by a repository, directory, database, or data included in flat files.
- **Management agent or call connector (MA)**: The **management agent** (**MA**) is the connector to a CD and manages the data specific to the connected data source. Currently, AD and Azure AD are the supported MAs, as you can see here: `https://docs.microsoft.com/en-us/microsoft-identity-manager/reference/microsoft-identity-manager-2016-connector-version-history`.
- **Connector space (CS)**: This represents a storage and staging area. It stores the states that indicates whether a piece of information has changed in the CD. Each CD has its logical sector in the CS.
- **Metaverse:** The central data store that contains the imported and aggregated identity information from all connected data sources. It provides a global view over all objects and attributes.
- **Staging:** If you run a staged import operation on an MA, such as Full/Delta Import (Stage Only), the data is imported from the **connected directory** (**CD**) into the CS, but no synchronization rule is applied to it. So, a staged import doesn't affect the metaverse.

- **Import:** The process of how objects and attributes from the connected data source will be moved into the connector space, including the associated operations, such as creation, modification, deletion, or verification. The import process can be a full or delta import.
- **Synchronization:** The process of applying all the configured rules to the staged objects in the connector space. Synchronization can be divided into inbound and outbound processes.
- **Export:** The process of writing changes that occurred during synchronization from the CS back to the connected data source.

Synchronization scenarios

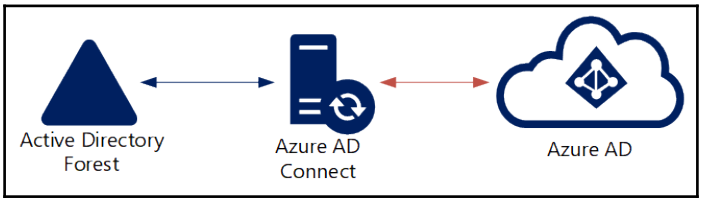
With the creation of a new Azure AD tenant, the directory information is managed independently from the on-premises AD forest by default. So, basically, a new onboarded user must be created in both directories: the Azure AD and the local AD. Unless you drive a cloud-only company, you always need to synchronize identities from the on-premises AD to the Azure AD tenant you own to provide a single identity. After the synchronization process is in place, Azure AD and AD can be viewed as one unique identity service. The following section provides you with several integration scenarios, including the user sign-in options. We will divide this section into the following situations:

- Single-forest integration
- Multi-forest integration
- Multi Azure Active Directory Integration
- Azure Active Directory Domain Services Integration
- Stretched Active Directory to Azure IaaS
- Azure Active Directory B2B Integration
- Azure Active Directory and Microsoft Office 365 synchronization
- Identity and password hash synchronization including SSO options
- Identity synchronization including PingFederate integration
- Identity and password hash synchronization including ADFS integration
- Azure Active Directory Connect high availability

Let's start with the single-forest integration.

Single-forest integration

The single-forest scenario is a commonly used one. A single forest can contain one or multiple domains and a single instance of **Azure AD**. The express settings of **Azure AD Connect** support this scenario. We recommend filtering the objects so that service accounts, computers, or other objects won't be synchronized to the cloud:



Azure AD Connect single-forest integration scenario

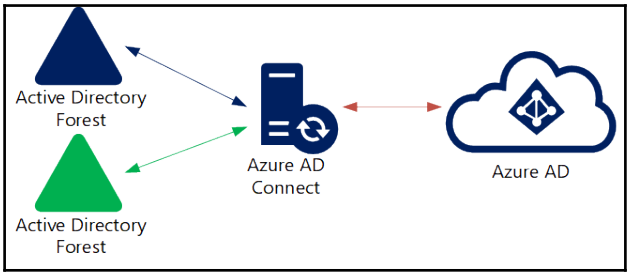
Additional **Azure AD Connect** servers connected to the same **Azure AD** aren't supported. Excluded is the high availability option with the **Azure AD Connect** staging mode, which is explained in the section, *Azure Active Directory Connect high availability*.

Multi-forest integration

Larger organizations or distributed organizations have environments with multiple on-premises ADs. They're typically used in account/resource forests or provided through mergers and acquisitions. These rules need to be followed:

- Users have only one enabled account across all on-premises **Active Directory Forests**
- `UserPrincipalName` and Source anchor will be provided from the forest
- Users have only one mailbox
- Users that have a linked mailbox also have an account in a different forest
- There's no need to use **Azure AD Connect** on a domain-joined server

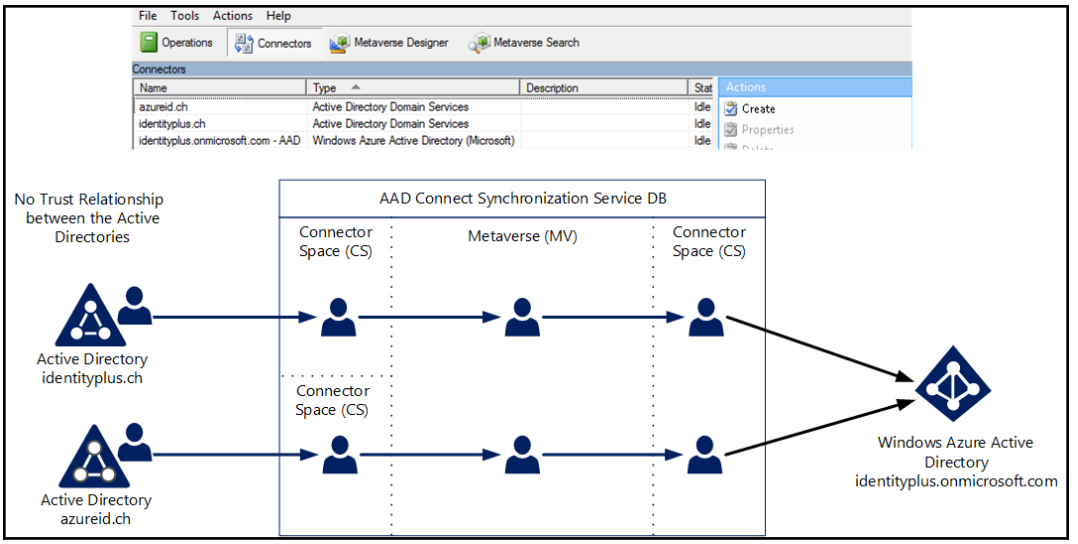
The following diagram shows the account/resource forest scenario:



Azure AD Connect multi-forest integration scenario

Remember: multiple forests and multiple **Azure AD Connect** tools on one Azure AD aren't supported. The only exception is the usage of a staging server. A staging server can be configured for high-availability scenarios (active/passive). In this scenario, the staging server doesn't export information to the target system.

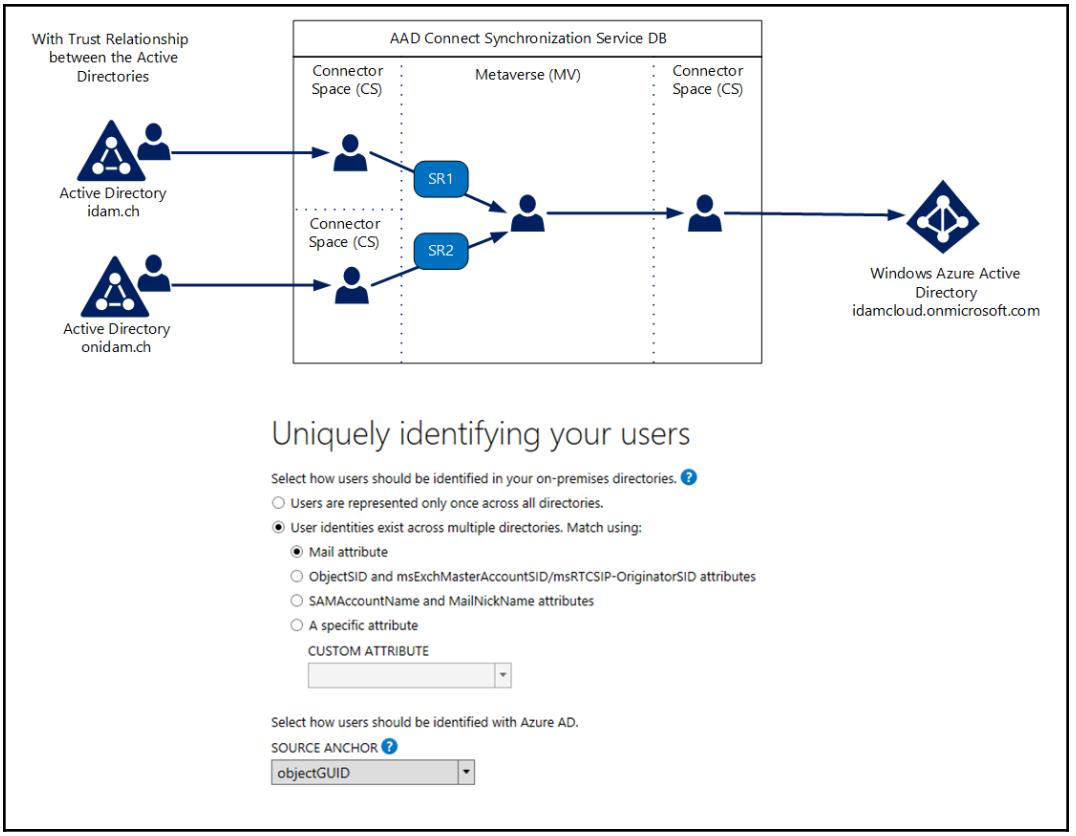
If you have multiple forests, a single sync server and the users are represented in one directory and, you choose the option **Users are represented only once across all directories**, then each object in every forest is represented once in the **Metaverse** and aggregated in the target Azure AD tenant:



Azure AD Connect multi-forest integration scenario including synchronization mappings

If you have numerous forests full of mesh with optional GALSync, you should use the **User identities exist across multiple directories** option.

If you use the mail attribute as matching criteria, the identity objects are joined with the mail attribute.This results in the following behavior—the user with a mailbox in one forest is joined with the contact in any other forest:



Multi-forest integration scenario (mail attribute mapping)

It's essential that objects are unique across AD forests. If objects are unique across every forest, you're in a good position. With object matching and joining, you can provide a good state when using cloud services. With the usage of joins, the precedence of synchronization rules also comes into play. If you join two objects based on an email address, and a specific attribute value where Object 1 isn't filled and Object 2 is populated, the value of the attribute (Object 2) will be used. But what if both objects have filled the attribute with different values? This is where precedence comes in.

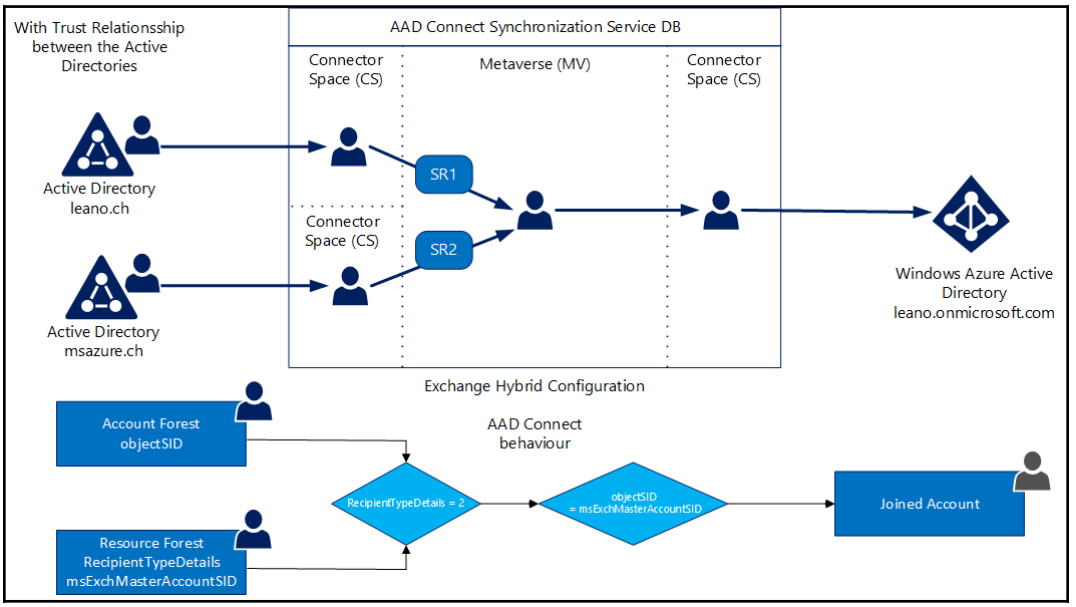
Rules with higher precedence are implemented later than lower valued rules.

The precedence will be set at the time of adding the forest to Azure AD Connect. So, if the forest of Object 1 was added before the forest of Object 2, the value of Object 1 will win the game. The preceding diagram shows this scenario.

Other objects are contacts you will find in such scenarios where a **Global Address List** (**GAL**) synchronization was implemented between two forests. Azure AD Connect provides the following default behaviors:

- If Azure AD contact finds a match of a contact and a user, a join will happen
- If there's no user object available, a contact object will be created
- If a subsequent user object is found with a match to a contact, a user object will be built in AAD
- If you have an account/resource forest scenario and you use the **User identities exist across multiple directories** option

If you use the matching criteria of **ObjectSID** and the **msExchangeMasterAccountSID** attribute, the expected results are that the users are disabled in this forest. Furthermore, the mailbox is linked to the account forest. The user will be presented uniquely in the Azure AD:



Account/Resource forest integration scenario

The preceding diagram also shows the Azure AD Connect behavior in a scenario where the AD object will be checked to determine whether it's a linked mailbox before it attempts to match **msExchMasterAccountSID.** This will be done with the **recipientTypeDetails** attribute. A value of `2` means that it's a linked mailbox. Keep in mind that disabled user accounts are also synchronized to Azure AD by default. Deactivated accounts are commonly used in exchange-resource forest deployments. The account forest holds the active user account and the resource forest holds the disabled user account. We will discuss rule precedence in `Chapter 3`, Exploring advanced synchronization concepts.

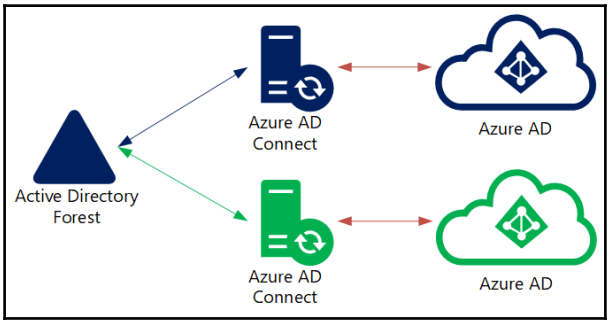
Multi-Azure Active Directory Integration

Sometimes you need to have multiple Azure Active Directories, for example if parts of your organization are based in China or you need to follow government regulations. For each Azure AD directory, you'll need one Azure AD Connect installation.

In a single-forest filtering scenario to multiple Azure ADs, the following needs to be done:

- Azure AD Connect must be configured for filtering
- DNS domain registration is only possible in a single Azure AD
- UPNs of the users on-premises must use separate namespaces
- Federation configuration needs to be customized
- One Azure AD directory can enable Exchange hybrid with the on-premises AD
- Global Address List synchronization needs to be performed through MIM 2016
- Windows 10 devices can only be with one Azure AD tenant
- The SSO option with the password hash synchronization and pass-through authentication activated can work only with one Azure AD tenant
- Group and device write-back scenarios are possible

The following diagram shows the multiple Azure AD situation:

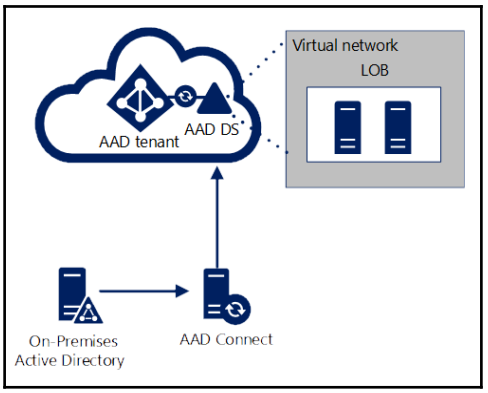


Connecting multiple Azure AD to one AD forest

It's unsupported to sync the same user to multiple Azure ADs.

Azure Active Directory Domain Services Integration

This service provides the capabilities to use AD Domain Services as a Service in Azure. It delivers two domain controllers with a small footprint of management options. It integrates directly with your Azure AD. It's an excellent option to move entirely to the cloud with everything you use on-premises. This provides smaller companies with an opportunity to live without a local infrastructure. Just imagine that a main legacy **LOB** can now be used in an Azure AD integration scenario and everything will be managed under service conditions:



Azure AD Domain Services integration scenario

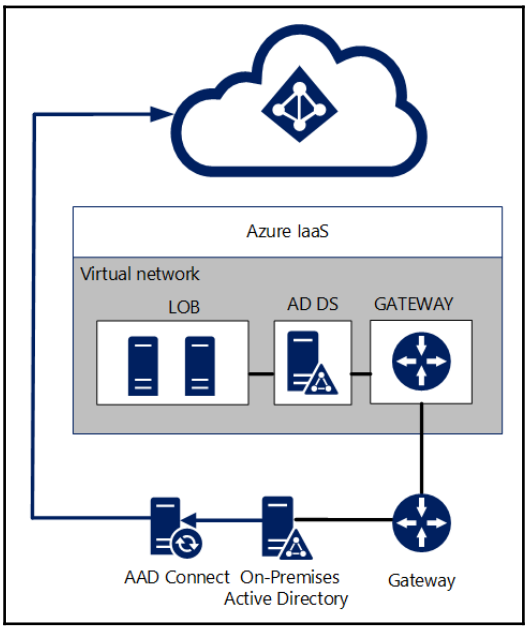
The solution provides capabilities to support NTLM/Kerberos and LDAP applications. You can also securely expose LDAP in an example for printing solutions. We used this option in `Chapter 1`, *Building and Managing Azure Active Directory*. Note that you need to activate the password-hash sync option in Azure AD Connect, so the user can be successfully synced between Azure AD and the Azure AD Domain Services. Furthermore, you benefit from the following additional features:

- **AD account lockout protection**—Users are locked out for 30 minutes if five invalid passwords are used within 2 minutes. Accounts are automatically unlocked after 30 minutes.
- **Custom organizational units** (**OUs**)—You are able to create multiple OUs.
- **Group policy support**—You are able to use group policies to manage servers.

Stretched Active Directory to Azure IaaS

Extending your local AD Domain Services to Azure IaaS provides you with a very flexible scenario to use your line-of-business applications in the cloud. To use this integration, you need to build a VPN or Express Route connection to Azure.

Domain controllers are highly sensitive roles and will have the most concerns focus on the trust of the service. Many alternative solutions don't support seamless lift and shift migration to Azure like this one:



Extending your Active Directory to the cloud

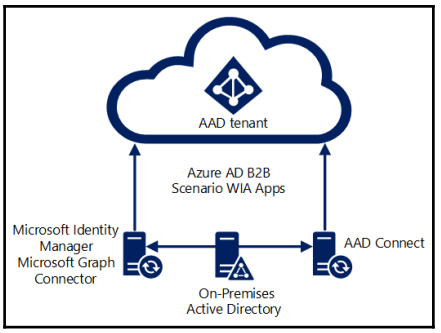
Follow the following notes:

- **Domain controllers (RW)**: The best choice for IaaS workloads and will be aware of your replication considerations
- **Domain controllers (RO)**: Normally used for scenarios with poor security and not an appropriate choice for IaaS workloads
- **Resource forest scenarios**: Not recommended for use in IaaS

In the next section, we'll take a look at the Azure AD B2B integration scenario.

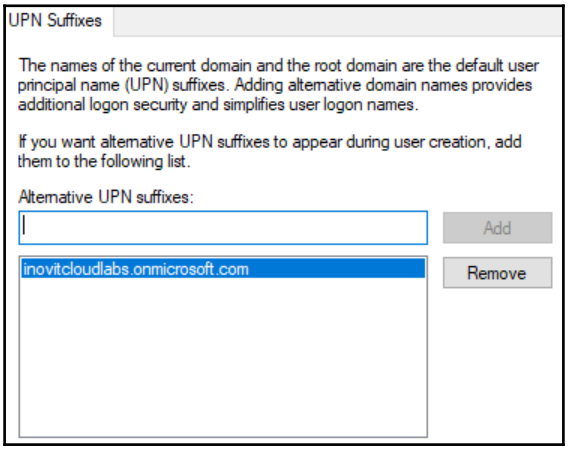
Azure Active Directory B2B integration

Azure AD B2B allows any partner to use their own identities and credentials in a collaboration scenario. For the authentication flows and capabilities, we'll take a more in-depth look in `Chapter 8`, *Using the Azure AD App Proxy and the Web Application Proxy* and `Chapter 10`, *Exploring Azure AD Identity Services*. For now, we'll give a quick overview of the synchronization part:



Azure AD B2B local application access scenario

For **Azure AD B2B** users to use on-premises Kerberos applications, we need to synchronize the guest user accounts back to the **On-Premises Active Directory**. For this reason, you need to provide your default Azure AD domain suffix in your local AD. In our case, it's `inovitcloudlabs.onmicrosoft.com`. You will find the option in the AD Domains and Trusts console:



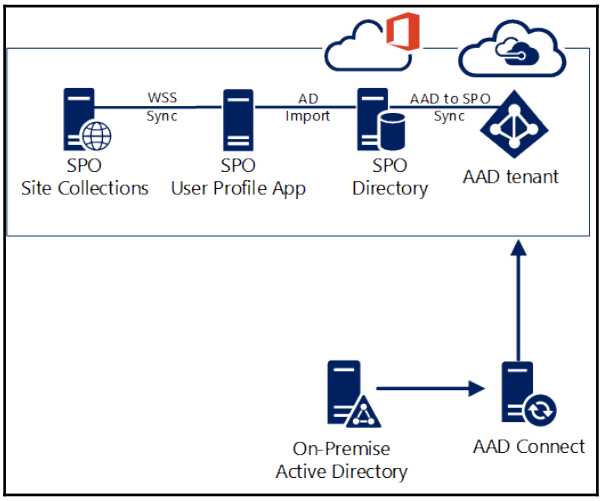
Adding your Azure AD tenant suffix to your local UPN suffixes

The registration of the new UPN suffix is necessary because the Azure AD Application Proxy checks the local AD for the existence of the guest user `UserPrincipalName`, such as `jochen.nickel_inovit.ch#EXT#@inovitcloudlabs.onmicrosoft.com`.

Microsoft provides a default solution for synchronizing the guest users back to the local AD; check it out at `https://bit.ly/2Bor7xy`. The solution contains the needed MIM 2016 configuration or a script to deploy the solution successfully. Don't be worried, we'll do the default configuration and extension later in the book.

Azure Active Directory and Microsoft Office 365 synchronization

The following scenario works for the most Office 365 services because every service uses its own directory inside of the Azure AD to store and manage identities. In particular, with SharePoint, we need many additional attributes in the **User Profile Application**, but you can't configure the sync between Azure AD and SharePoint Online. You can find all the default attributes at `https://support.office.com/en-us/article/information-about-user-profile-synchronization-in-sharepoint-online-177eb196-5887-43c9-84c3-b98a43d35129`. The following diagram shows the SharePoint Online synchronization scenario in a schematic view:



Azure AD to User Profile App synchronization extension

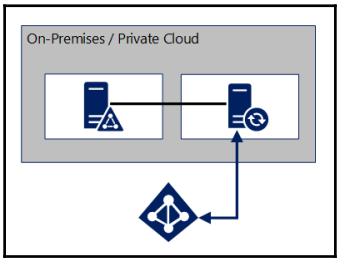
One option is to extend the synchronization part with your solution, using the Microsoft Azure AD Graph API. In our case, we used a complete serverless solution based on Azure Functions and Logic Apps.

Identity and password-hash synchronization including SSO options

By synchronizing identities and the associated password hashes from the on-premises AD to the Azure AD, we can build a basic scenario for smaller companies that don't want to invest in an ADFS infrastructure. Also, there's no SSO required. With this scenario, the same password can be used to authenticate the user either in the cloud or on-premises, depending on what resource is being accessed. Furthermore, the Password Reset and Account Unlock features are available with an Azure AD Premium license. A requirement is Azure AD Connect with password-hash synchronization enabled. Optional password write-back is enabled.

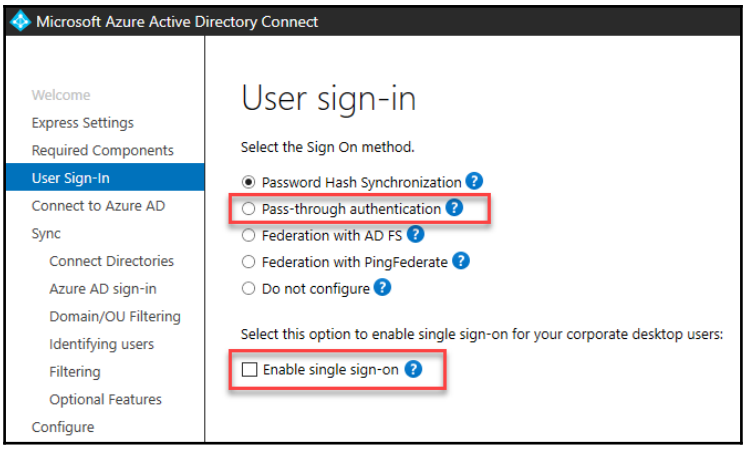
For this process, a rehashing functionality is in place, which allows the user to have two different hash values in the local AD and the Azure AD. Additionally, multi-forest synchronization is also supported.

The following diagram shows the identity and password-hash synchronization scenario:



Azure AD Connect password-hash synchronization scenario

To add SSO to the solution, you can enable **Pass-through authentication** and the seamless SSO feature in the Azure AD Connect tool. This is the most commonly recommended option from Microsoft to reduce complexity and put Azure AD in the role of the central system to provide authentication to your SaaS and on-premises Kerberos/Claims-based applications:

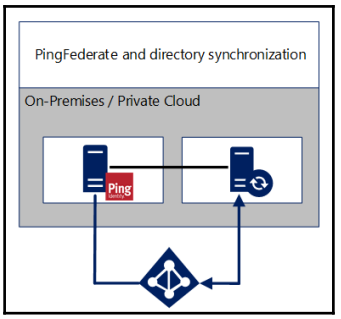


PTA and seamless SSO enablement

It's highly recommended you enable password-hash synchronization, so in case of an on-premises service interrupt, your users can still use cloud services. For now, you can read about this feature at `https://docs.microsoft.com/en-us/azure/active-directory/hybrid/how-to-connect-pta`.

Identity synchronization including PingFederate integration

Microsoft now directly integrates non-Microsoft products into Azure AD Connect. The first one is the integration of PingFederate, which is very popular in the market. There's no change on the directory synchronization—you enable PingFederate as your federation provider instead of ADFS:



Integrating PingIdentity for federation and header-based authentication

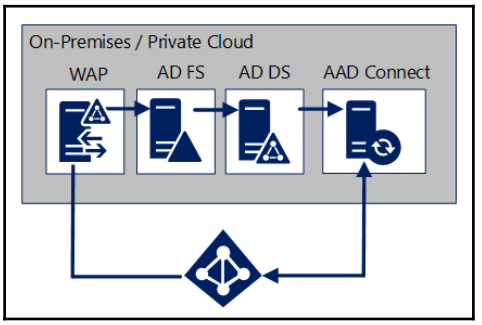
The main decision that Microsoft has made with this partnership is to move more and more into the hybrid identity management approach to help you to enable Microsoft and other cloud services quickly.

Identity and password-hash synchronization including ADFS integration

With the implementation of the federation, all authentication is retained on-premises, and all passwords are stored on-premises only. All authentication traffic is redirected from Azure AD to the on-premises ADFS, which authenticates the user against a trusted AD domain. This scenario is commonly used in different company sizes if SSO is required and password-hash synchronization is prohibited due to \ security reasons.

The requirement is the usage of a federation service provider, such as ADFS in addition to Azure AD Connect in a highly available deployment.

The following diagram shows the identity and password-hash synchronization with **ADFS** scenario:



Combine federation with password-hash synchronization

You can also combine the **ADFS** integration with password-hash synchronization to provide the capability if the on-premises infrastructure turns into an outage and users can still access their cloud services with their known password.

Azure Active Directory Connect high availability

For high-availability reasons, a new Azure AD Connect server can be rebuilt and resynchronized in a couple of hours for a small or medium-sized business.

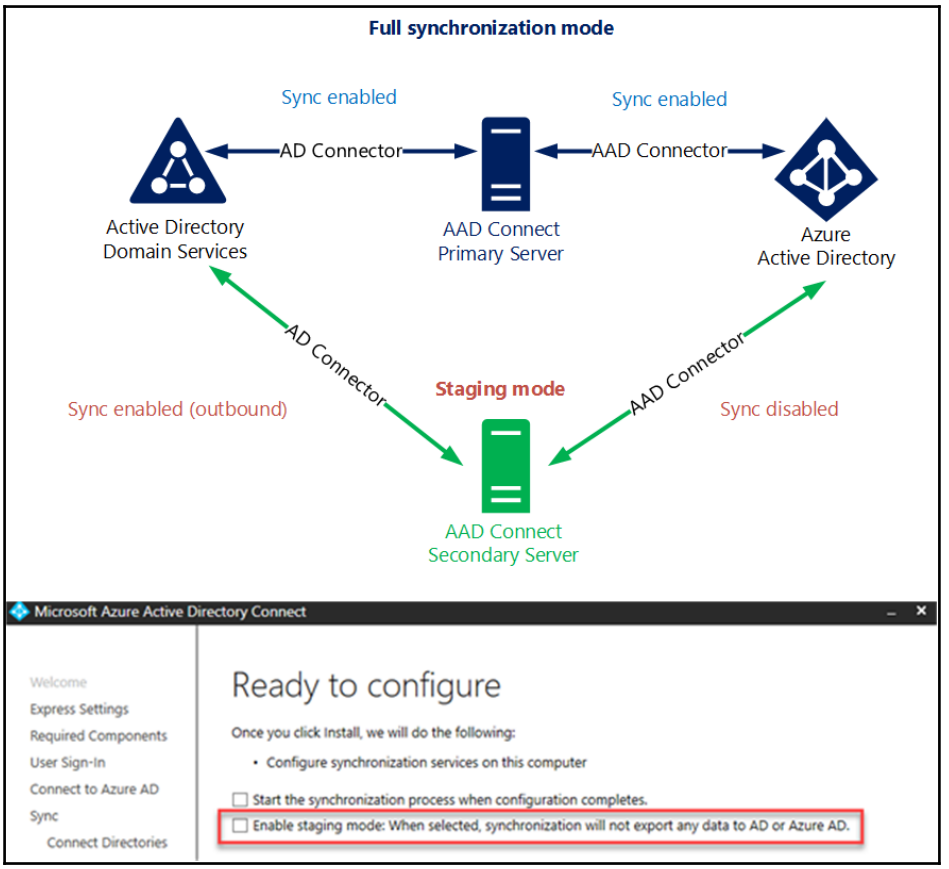
Remember that the `sourceAnchor` attribute is used to join the objects from on-premises and the cloud. The sync engine matches the objects together again on reinstallation.

It's very important that you document your configuration changes, such as special filtering or synchronization rules. You need to reapply these settings before you start the synchronization process. You can use the Azure AD Connect documenter (`https://github.com/Microsoft/AADConnectConfigDocumenter`) to save your changes.

Larger organizations with more than 100,000 users, groups, and other objects will take much more time to rebuild the synchronization. If there needs to be a faster time to recovery, Azure Active Directory Connect can be configured to use a dedicated SQL server deployment with SQL high availability. This option provides the needed data directly. With more than 100,000 users, an SQL server is required because a large organization wants to have low recovery time for the synchronization service.

Another way to provide a highly redundant system is to use another server with Azure Active Directory Connect installed and configured in **Staging mode**. This functionality also reduces recovery time.

The following diagram shows a staging-mode configuration:



Enable the staging mode for HA

With Azure AD Connect version 1.1.524.0, Microsoft added the SQL **Always-on Availability** (**AOA**) Group support. SQL clustering was added in an earlier release. Be aware that SQL AOA needs to be enabled before you install Azure AD Connect.

Synchronization terms and processes

In this section, we'll discuss and implement the practical use of the synchronization terms and procedures. We'll combine theory directly with practical use. For this reason, we'll install, configure, and run the processes immediately in the Azure AD Connect tool. To use the guidance, you should deploy a virtual machine with the domain controller role enabled.

Build the virtual machine on Azure or your local virtualization platform. An excellent option is to follow the guide at `https://docs.microsoft.com/en-us/office365/enterprise/base-configuration-dev-test-environment` with the usage of your free trial Azure or MSDN subscription. We provide you with a complete scripting solution in the code package of the book, or you can follow the instructions in `Chapter 7`, *Deploying Solutions on Azure AD and ADFS*.

We use the same domain name you used in `Chapter 1`, *Building and Managing Azure Active Directory*. In our case, we use the domain name `inovitlabs.ch`. So, change the scripts for your environment.

Now that we have our primary test environment in place, we can start the preparation and installation of the Azure AD Connect on the Domain Controller. We use this scenario to reduce the costs of your test environment. Be aware that we'll extend the test environment in the coming chapters to demonstrate the functionalities we discuss in this book.

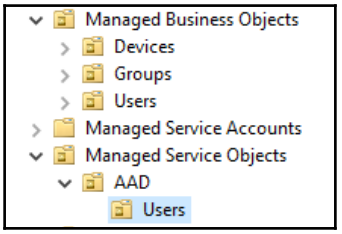
Are you ready? Let's prepare the domain:

1. Log in with the domain administrator credentials and run the following script to create the demo organizational unit structure:

```
New-ADOrganizationalUnit -Name "Managed Business Objects" -
Path "DC=INOVITLABS,DC=CH"

New-ADOrganizationalUnit -Name "Users" -Path "OU=Managed
Business Objects,DC=INOVITLABS,DC=CH"

New-ADOrganizationalUnit -Name "Groups" -Path "OU=Managed
Business Objects,DC=INOVITLABS,DC=CH"

New-ADOrganizationalUnit -Name "Devices" -Path "OU=Managed
Business Objects,DC=INOVITLABS,DC=CH"

New-ADOrganizationalUnit -Name "Managed Service Objects" -Path
"DC=INOVITLABS,DC=CH"
```

```
New-ADOrganizationalUnit -Name "AAD" -Path "OU=Managed Service
Objects,DC=INOVITLABS,DC=CH"

New-ADOrganizationalUnit -Name "Users" -Path
"OU=AAD,OU=Managed Service Objects,DC=INOVITLABS,DC=CH"
```

The following diagram shows the expected result:



Azure AD service organizational unit

2. Enable the Active Directory recycle bin feature:

```
Enable-ADOptionalFeature -Identity 'CN=Recycle Bin
Feature,CN=Optional Features,CN=Directory Service,CN=Windows
NT,CN=Services,CN=Configuration,DC=inovitlabs,DC=ch' -Scope
ForestOrConfigurationSet -Target 'inovitlabs.ch'
```

3. Create the **group-managed service account** (**gMSA**) to run the Azure AD Connect service. Replace the computer name with the one you choose for your test environment:

```
Add-KdsRootKey -EffectiveTime (Get-Date).AddHours(-10)

New-ADServiceAccount -Name svcaadconnect -DNSHostname
INOLABSADS01 -PrincipalsAllowedToRetrieveManagedPassword
INOLABSADS01$
```

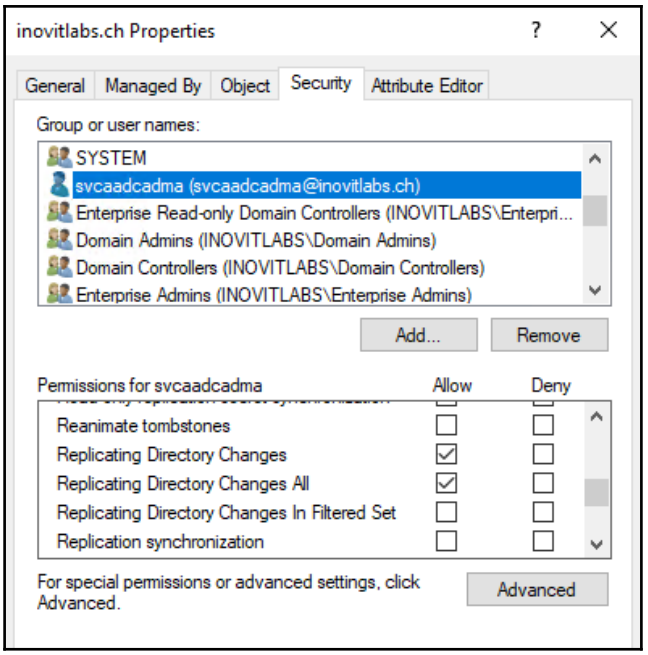
4. Create the service account for the Active Directory Management Agent that will be used to connect and do the synchronization operations:

```
New-ADUser -Name "svcaadcadma" -SamAccountName svcaadcadma -
UserPrincipalName svcaadcadma@inovitlabs.ch -path
"OU=Users,OU=AAD,OU=Managed Service
Objects,DC=inovitlabs,DC=ch" -AccountPassword (ConvertTo-
SecureString "Pass@word1" -AsPlainText -Force) -Enabled $True
```

The Active Directory Management Agent account needs to be configured with the correct permissions on the domain level.

5. Configure the permissions to configure the `svcaadcadma` Azure AD Connect with the Active Directory user's and computer's console (`dsa.msc`). Don't forget to enable the advanced features under the view option where you can see the **Security** tabs:
    - **Replicate Directory Changes**
    - **Replicate Directory Changes All**

The following screenshot shows the expected result:

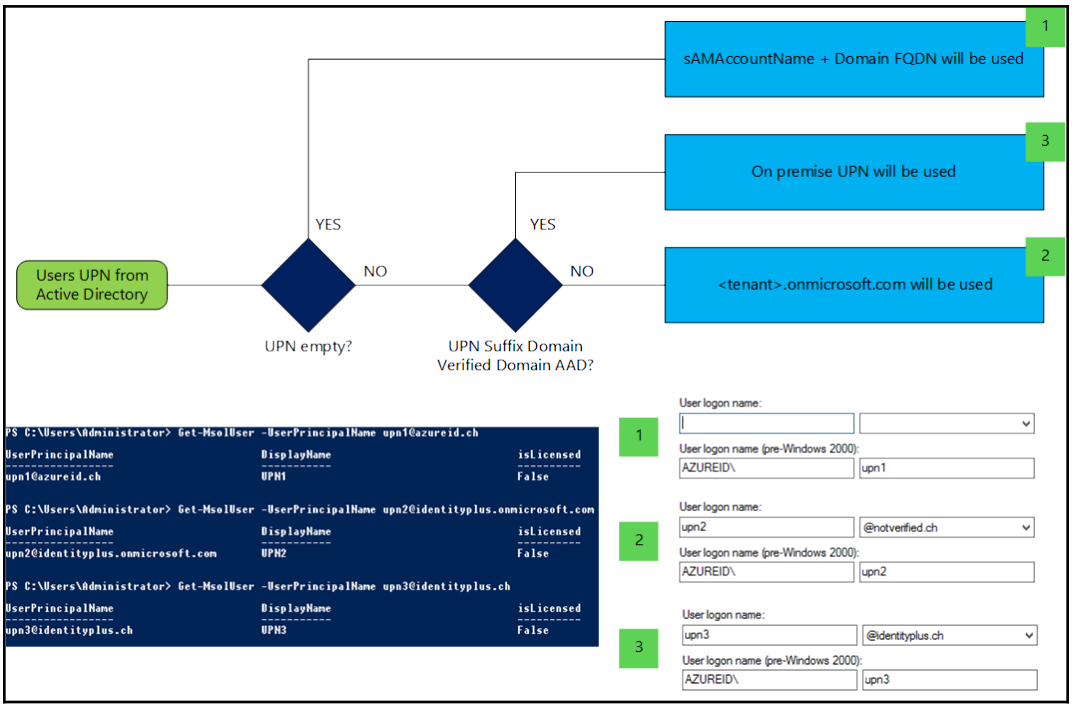


Assigning correct permissions to the Azure AD Connect AD Management Agent service account

Now that we've finished the preparation tasks in our test environment, let's run through the following sections for the theoretical explanations and the practical execution. For every task, we'll use the same credentials in an evaluated PowerShell session.

UserPrincipalName suffix decisions

`UserPrincipalName` is one of the most relevant user attributes in the connection from a local AD to the **Azure AD**. Azure AD Connect follows the rules shown here:



User Principal Name decision path

As you can see in the preceding diagram, AAD Connect uses the following logic by default:

- If a UPN is available, it will be used
- If a UPN isn't available, it uses the user's **sAMAccountName** and the **fully qualified domain name** (**FQDN**) of the connect AD domain
- If the UPN to be exported to the Azure AD isn't verified, the suffix will be replaced with **<tenant>.onmicrosoft.com**

Remember that users can exist in any synchronized forest and they include `UserPrincipalName` and `sourceAnchor`. In the case of using linked mailboxes, they will be ignored because the synchronization engine will find an active and deactivated account representation of the user.

Active Directory preparations

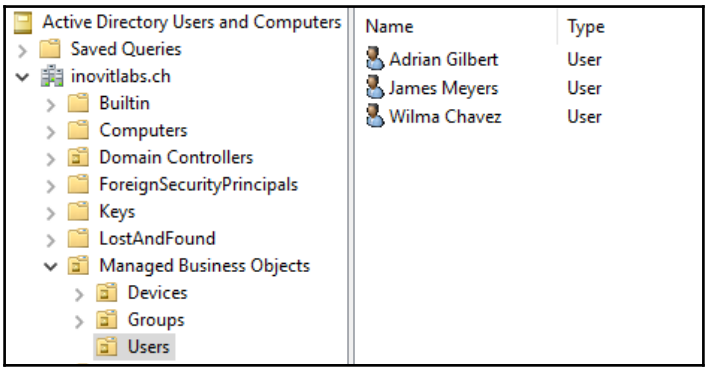
To prepare your AD environment, you can use the IdFix tool, which you can download from `http://bit.ly/1VnsvVn`. It performs the discovery and remediation of identity objects and their attributes in an on-premises AD environment in preparation for synchronization to Azure AD. IdFix is provided for AD administrators that plan to use Azure AD Connect with the Azure AD/Office 365 services. You can use the tool for every synchronization scenario:

Incorrect user accounts found by the IdFix tool

To test the **IdFix** utility, we'll create some incorrect test users with the following script:

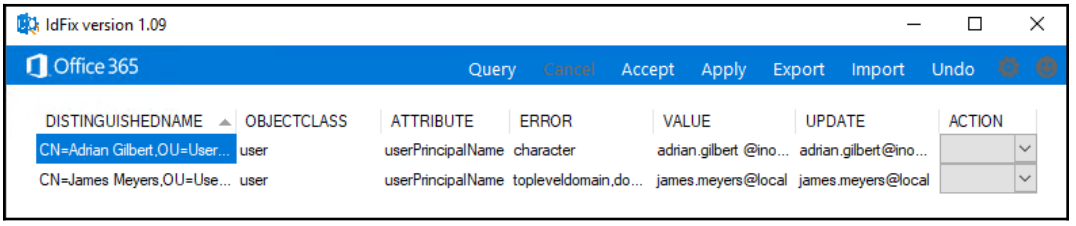
```
New-ADUser -Name "James Meyers" -GivenName J. -Surname Meyers -
SamAccountName jmeyers -UserPrincipalName james.meyers@local -path
"OU=Users,OU=Managed Business Objects,DC=inovitlabs,DC=ch" -
AccountPassword (ConvertTo-SecureString "Pass@word1" -AsPlainText -
Force) -Enabled $True

New-ADUser -Name "Adrian Gilbert" -GivenName Adrian -Surname Gilbert -
SamAccountName adrian.gilbert -UserPrincipalName "adrian.gilbert
@inovitlabs.ch" -path "OU=Users,OU=Managed Business
Objects,DC=inovitlabs,DC=ch" -AccountPassword (ConvertTo-SecureString
"Pass@word1" -AsPlainText -Force) -Enabled $True

New-ADUser -Name "Wilma Chavez" -GivenName Wilma -Surname Chavez -
SamAccountName wilma.chavez -UserPrincipalName
"wilma.chavez°@inovitlabs.ch" -path "OU=Users,OU=Managed Business
Objects,DC=inovitlabs,DC=ch" -AccountPassword (ConvertTo-SecureString
"Pass@word1" -AsPlainText -Force) -Enabled $True
```

The following screenshot shows the expected result:



Creation result of the damaged user accounts

Now we can run the **IdFix** tool to check the local AD for a user that will build errors in a synchronization:

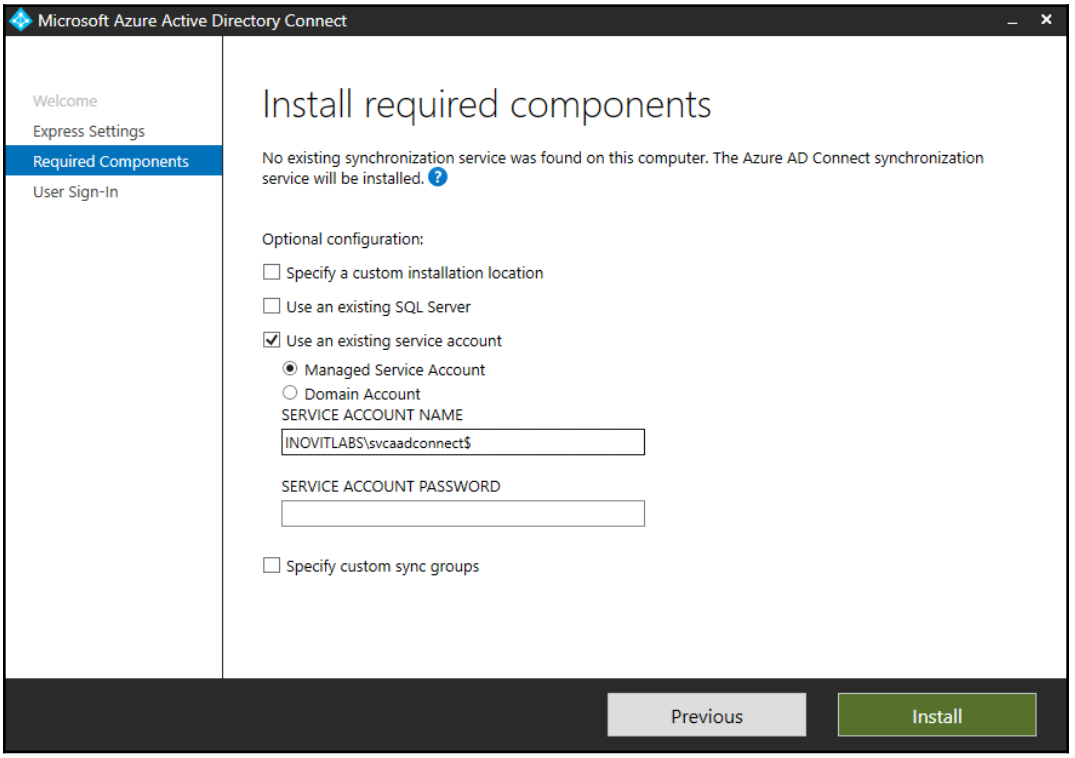


Errors found on the damaged user accounts

After you test the **IdFix** tool, delete the created test user accounts.

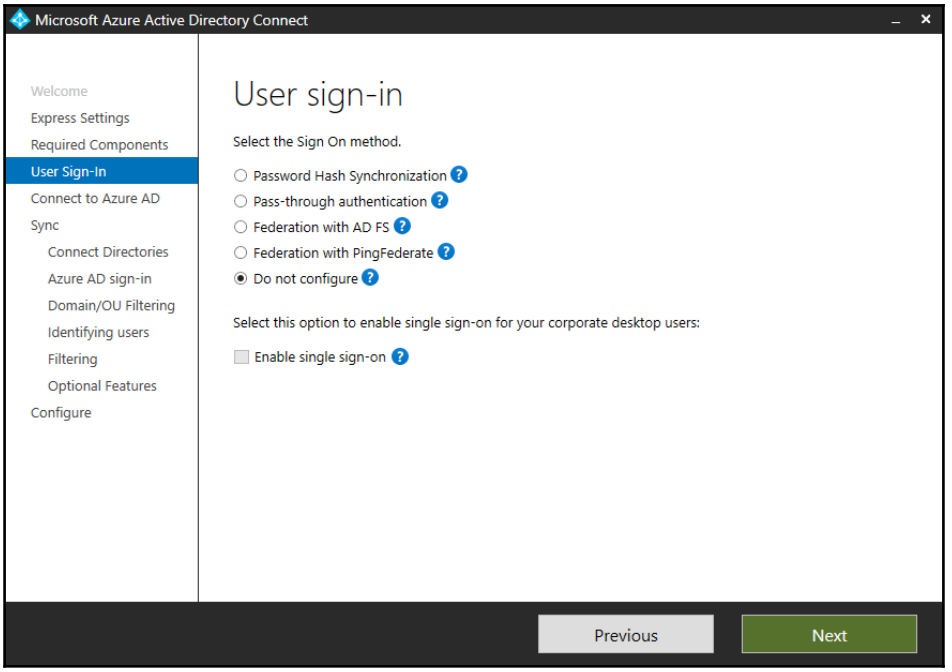
With the next steps we will start the installation of the Azure AD Connect tool:

1. Run the Azure AD Connect installation. Download the actual version of the tool from `https://www.microsoft.com/en-us/download/details.aspx?id=47594` and start the installation with the Domain Administrator credentials.
2. Choose the custom installation option so that we can view all the essential configuration steps. I always use the custom option and not the **Express** option.
3. Use the gMSA created in the previous steps to configure the Azure AD Connect service:



Service account configuration

4. At this time, we don't set any **User sign-in** option:



User sign-in configuration

In the next section, we'll discuss the source anchor decision process, so click **Next** and wait for the next lab part.

Source Anchor decisions

It's crucial to understand Source Anchor because it builds the basis for the relationship between the AD Domain Services user and the Azure AD user. The `sourceAnchor` attribute can't be changed. Be aware that your configuration fits your expected scenario.

The attribute provides the following capabilities to you:

- Supports faster rebuilding scenarios
- Supports the move from a cloud-only to a synchronized scenario with a hard match

- In a federation scenario, it builds the claim with `UserPrincipalName` to identify a user uniquely

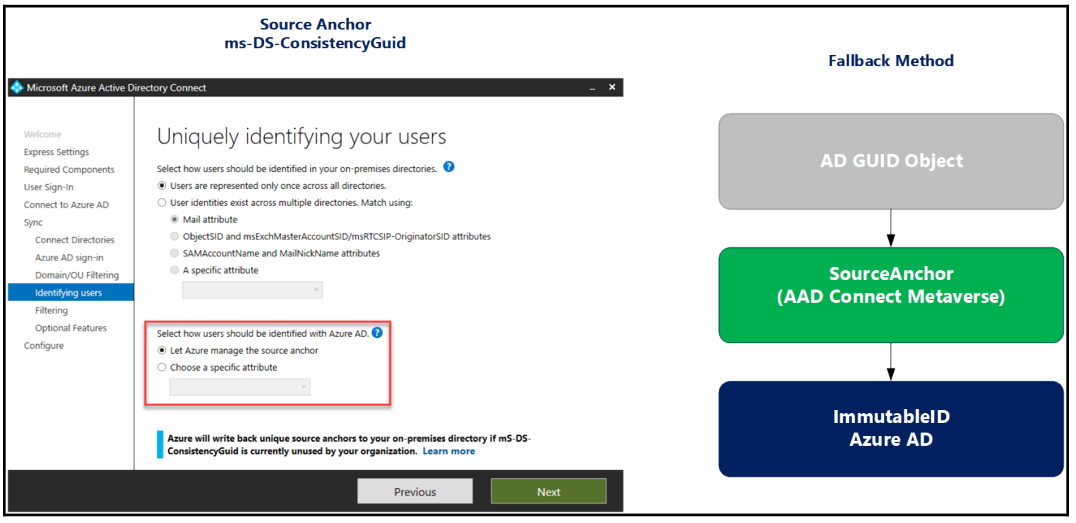
The following rules need to be followed when choosing a `sourceAnchor` attribute:

- Less than 60 characters
- No special characters, such as {, }, |, ~, <, >, (,), ' ; : , [,], ", @, _, \, !, #, $, %, &, *, +, /, =, ?, ^, or `
- Globally unique
- String, integer, or binary
- Should not be based on the user's name and case-sensitive
- Assigned at object-creation time

Azure AD Connect uses the following defaults as the `sourceAnchor` attribute:

- Azure AD Connect (version 1.1.486.0 and older) usage of the objectGUID as the `sourceAnchor` attribute
- Azure AD Connect (version 1.1.524.0 and after) usage of the **ms-DS-ConsistencyGuid** as `sourceAnchor` attribute

The following screenshot shows the configuration in the Azure AD Connect installation wizard:

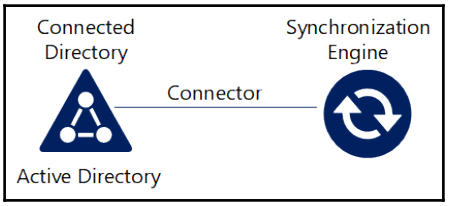


sourceAnchor definition of Azure AD Connect

The Management Agent user account must be granted write permissions to the **ms-DS-ConsistencyGuid** attribute in the local AD. Keep in mind, only newer versions of Azure AD Connect (1.1.552.0 and after) support switching from ObjectGuid to **ms-DS-ConsistencyGuid** as the `sourceAnchor` attribute.

Connected Directories

The **Synchronization Engine** from the Azure AD Connect processes identities from the AD and the Azure AD. Every directory or repository organizes the data in their method and provides different access methods. The data repositories that are synchronized with Azure AD Connect are called connected data sources or **connected directories** (**CDs**). The two supported CDs in Azure AD Connect are the AD and the Azure AD. You can connect any other repository but in a Microsoft, unsupported way. The following diagram shows the basic concept of synchronization:

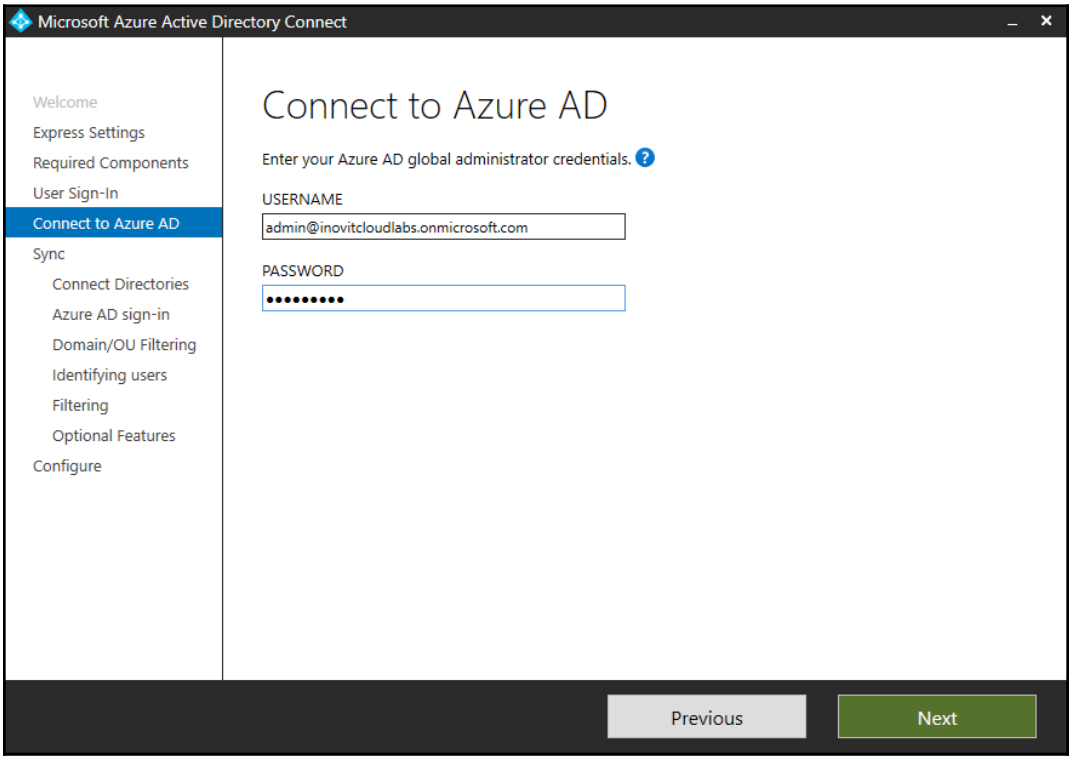


Basic concept of a connector and the connected directory

Now, let's jump back to the Azure AD Connect configuration and log back into your Domain Controller.

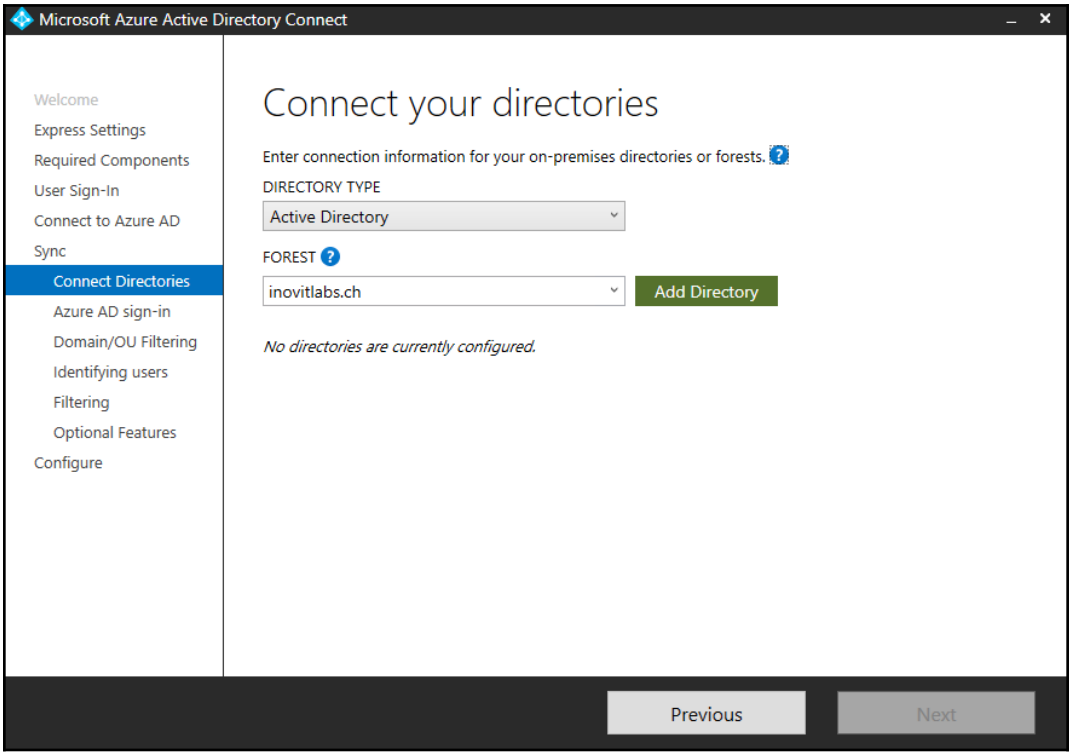
We will configure the two repositories:

1. Provide your **global administrator credentials** from your **Azure AD** and click **Next**:



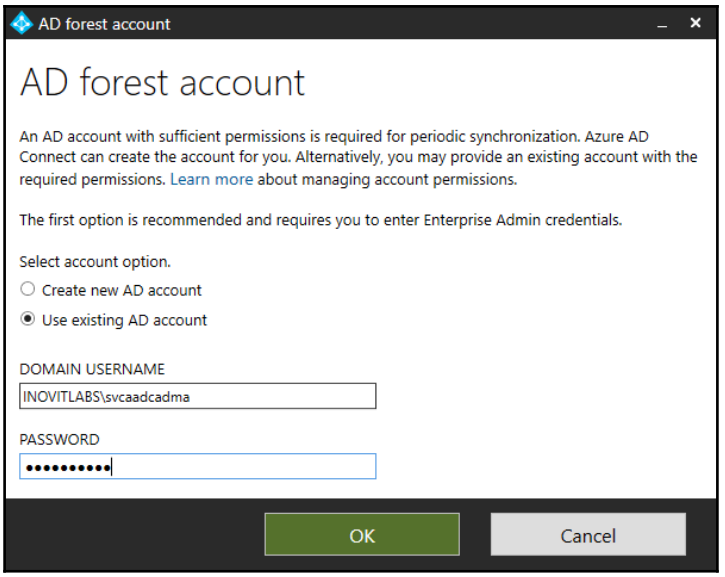
Connect Azure AD Connect to the Azure AD

2. Connect the local **Active Directory**:



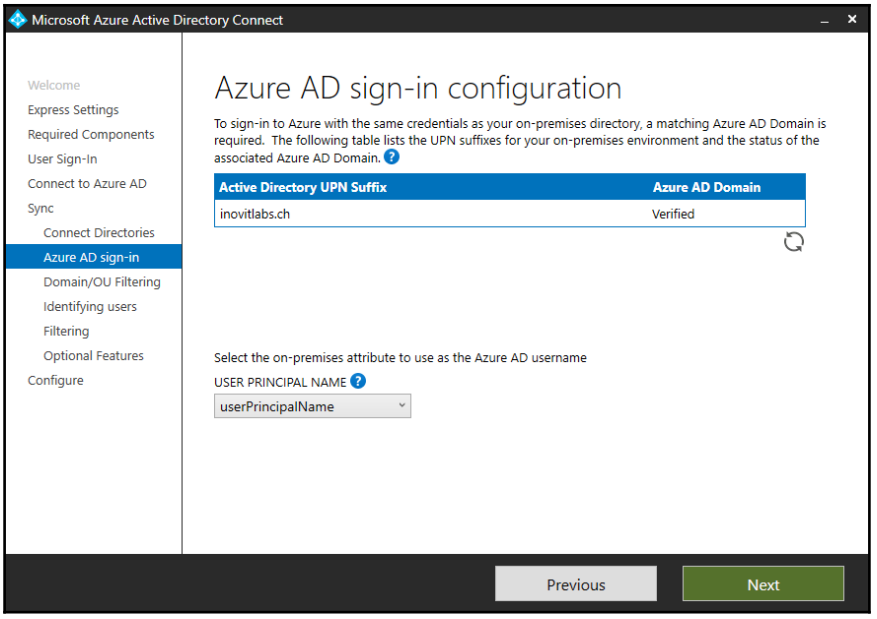
Connect the local AD

3. Use the created AD Management Agent account for this task:



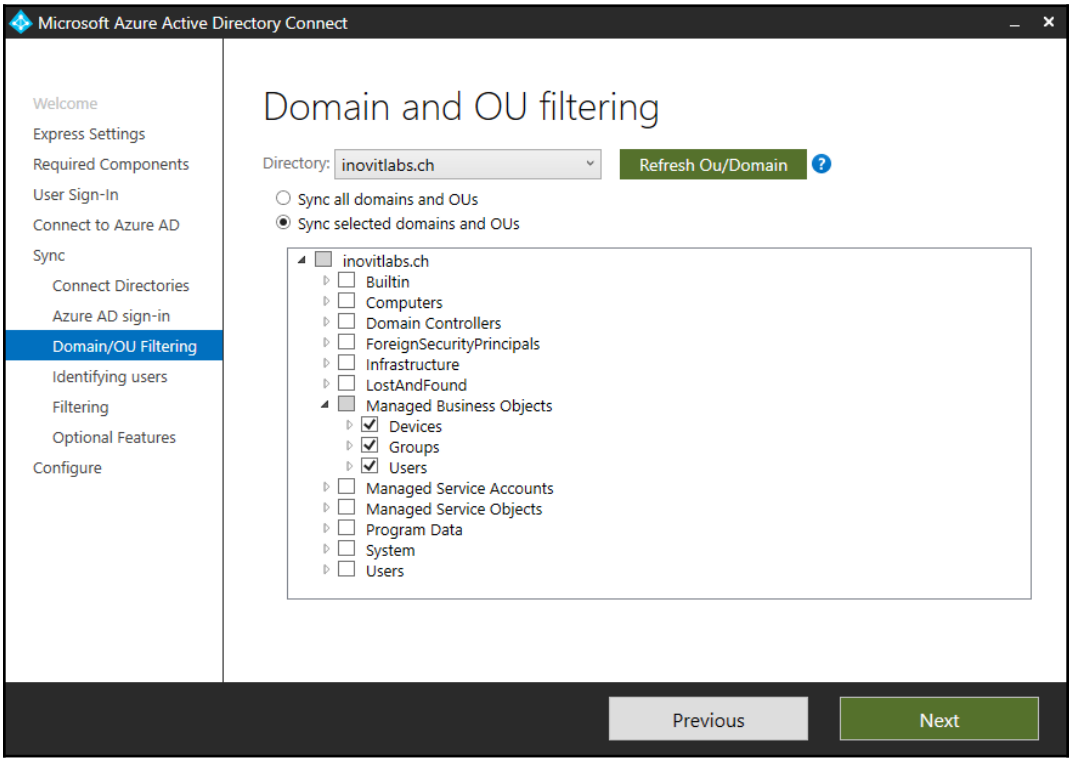
Configure the AD Management Agent service account

4. Find your verified custom domain, in my case **inovitlabs.ch**:



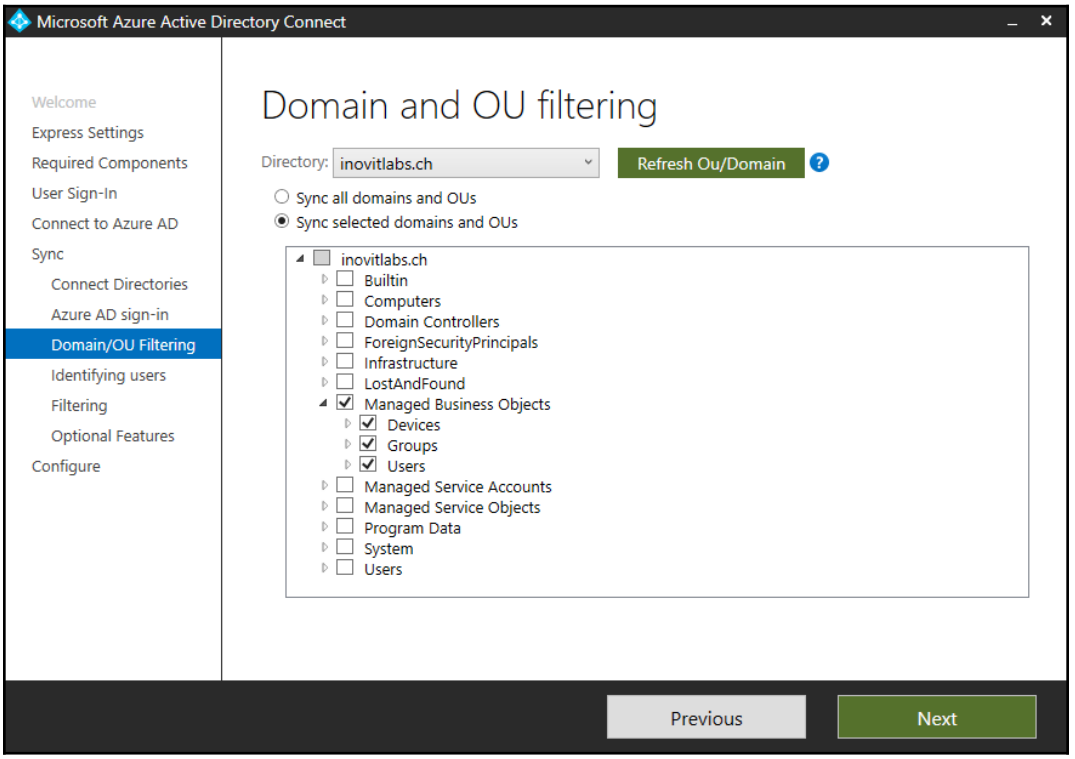
Define the sign-in options

5. On the **Domain and OU filtering** page, choose the created **Managed Business Objects** OU. With the following setting, no newly created OUs will be included in the synchronization process:



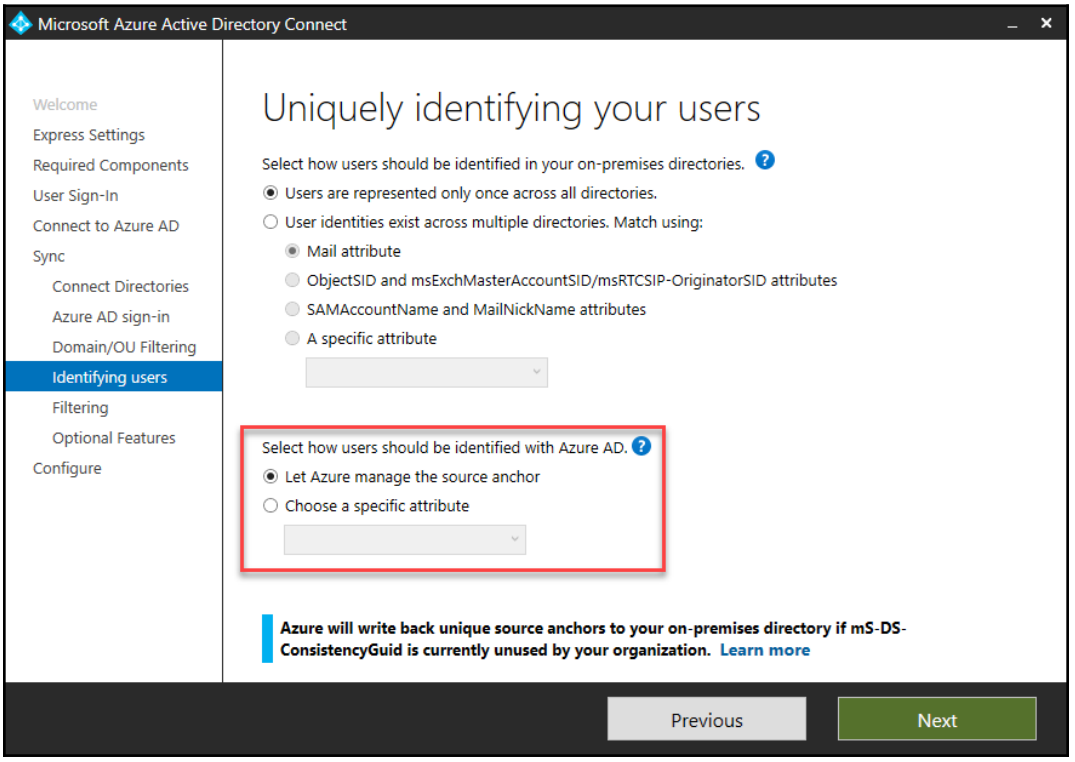
OU filtering options

6. With the following option, every newly established OU under **Managed Business Objects** will be included in the synchronization process:



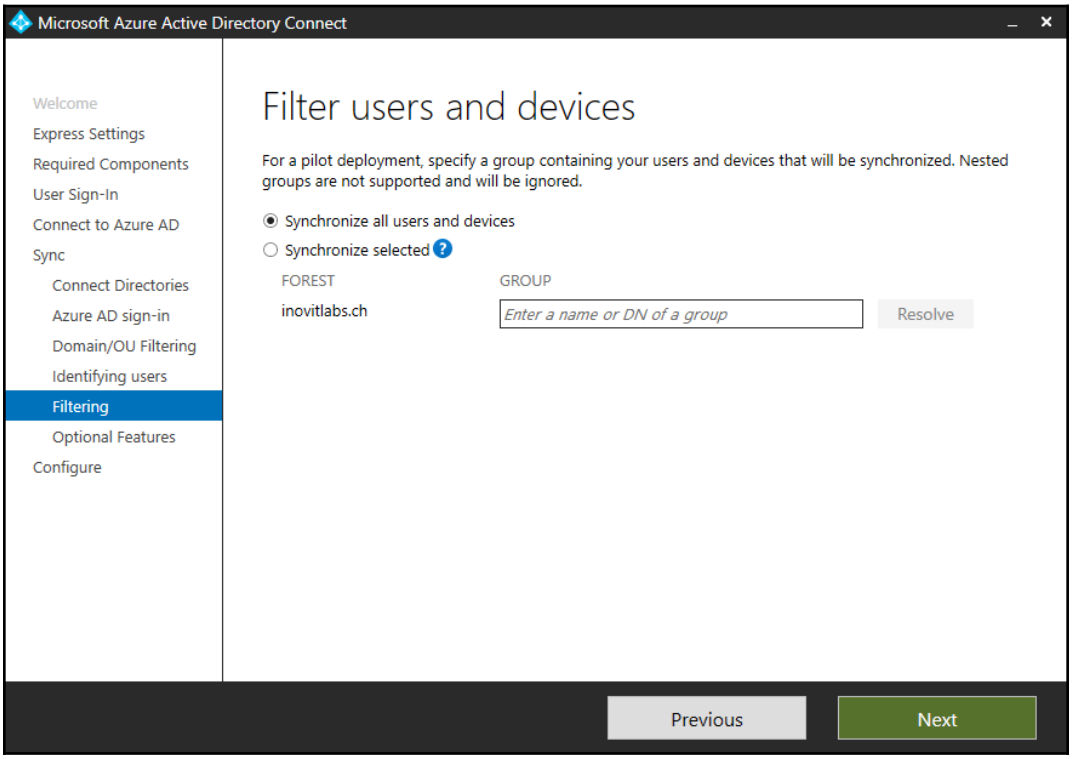
The second part of OU filtering options

7. Configure the `sourceAnchor` **ms-DS-ConsistencyGuid** as we discussed in the *Source Anchor decisions* section :



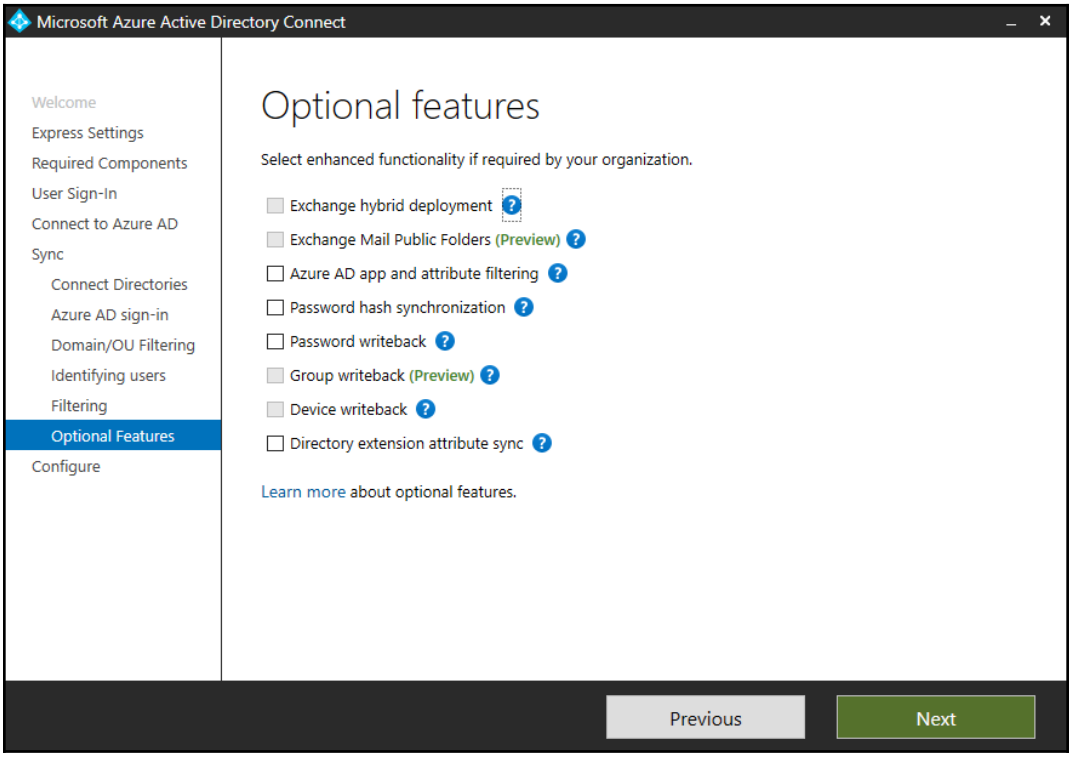
sourceAnchor definition dialog

8. In the **Filter users and devices** section, choose the default option. You could also use the group option for proof of concepts:



Filter users and devices dialog

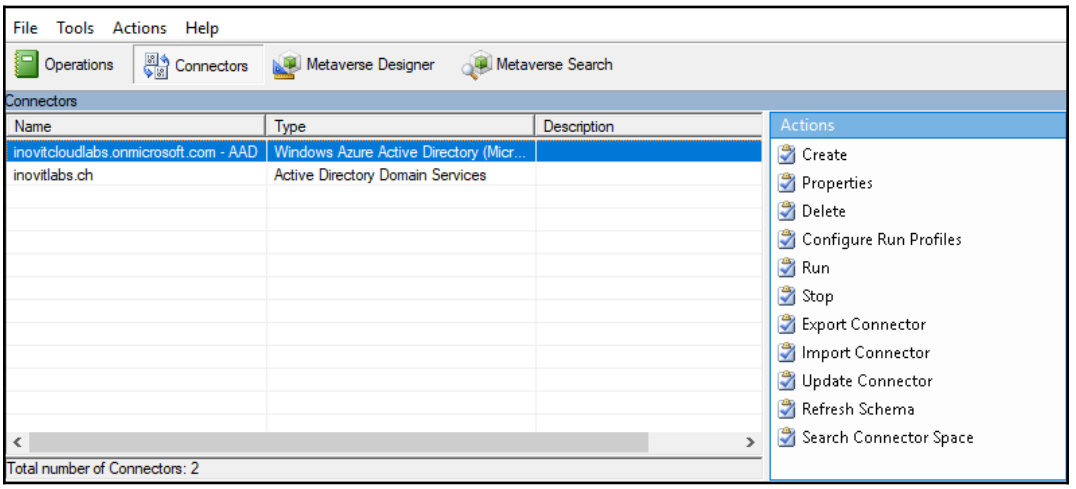
9. In the **Optional features** section, you see the available synchronization options, which we discussed in the integration scenarios. Keep in mind that you need the correct permissions for the Active Directory Management Agent account to write back to the local AD:



Azure AD Connect Optional features section

10. Click **Next** to finish the configuration. Be sure that you don't enable the synchronization process because we do the things step by step.

The directories will be connected after we finish the Azure AD Connect installation wizard, and can be viewed in the **Synchronization Service Manager** under **Connectors**:



Azure AD Connect - Connected directories

After installing and configuring the Azure AD Connect tool, we can jump into the detailed sections of the synchronization. In the upcoming `Chapter 3`, *Exploring Advanced Synchronization Concepts*, we'll enable the available writeback, extension, and filtering options.

Import flow

The internal sync engine contains two namespaces that store identity information: the CS and the **metaverse** (**MV**). The CS is used to detect changes in the connected data sources. The connector space is also used to proceed outgoing changes for the export process into the connected data source.

Data can flow in each direction, but it can't flow in both directions simultaneously.
Just to recap: the MV is a storage area that contains the aggregated identity information from multiple connected data. It's very important to understand that every synchronization engine object must have a **globally unique identifier** (**GUID**) for data integration reasons and the relationships between the objects.

The import process is done between the connected data source and the connector space. For this reason, we provide some information about the connector space.

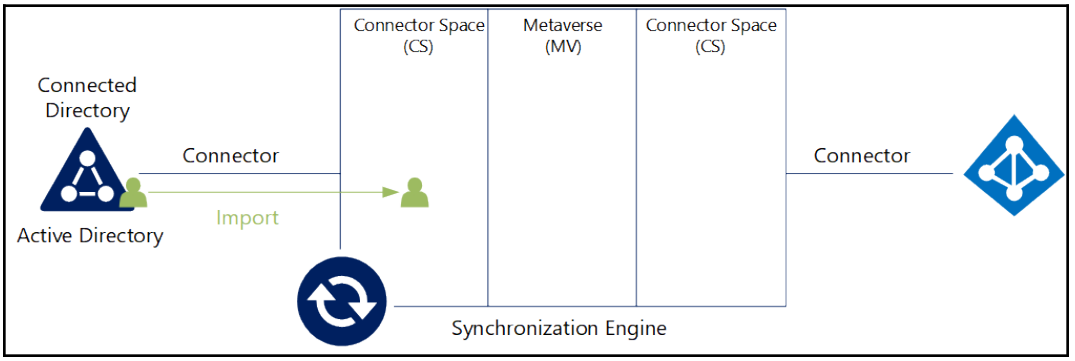
Connector space objects have two attributes:

- **Globally unique identifier** (**GUID**)
- **Distinguished name** (**DN**)

CS objects can be one of the following:

- A CS object
- A placeholder

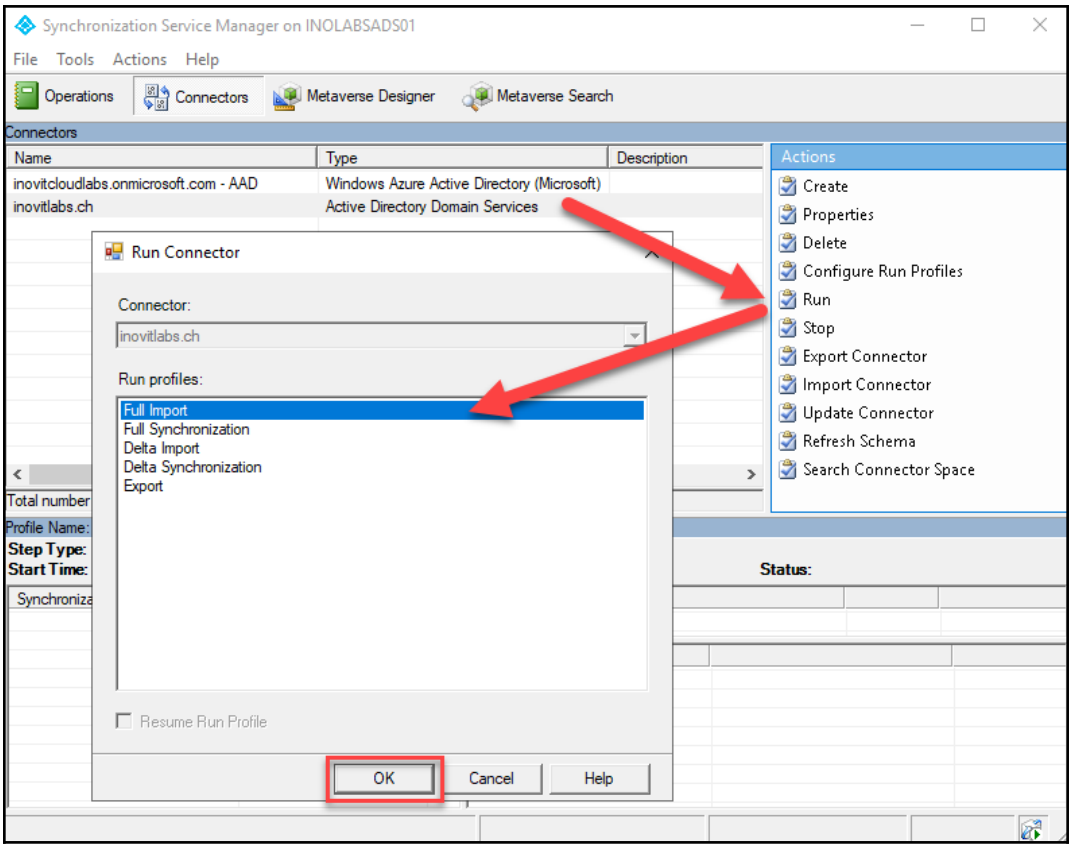
The following diagram shows the schematic view of the **Synchronization Engine**:



Synchronization Engine overview

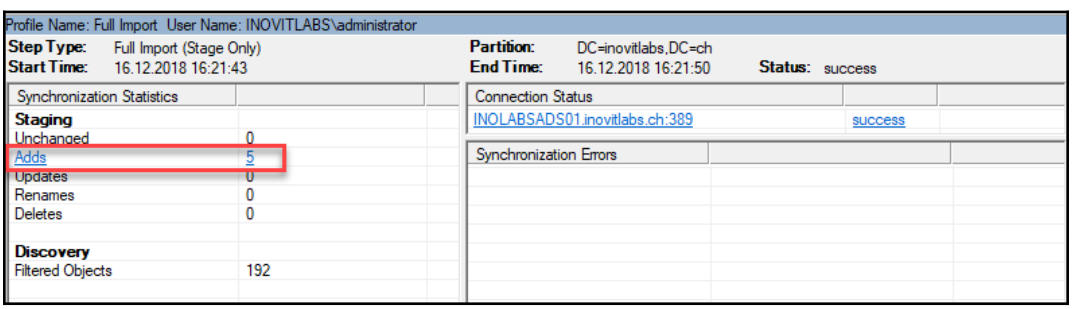
Now let's see the import flow in action. Keep in mind that we already have users in our Azure AD from `Chapter 1`, *Building and Managing Azure AD:*

1. Log into your Domain Controller and open the Azure AD Connect **Synchronization Service Manager.**
2. Run a **Full Import** on the connected Active Directory:



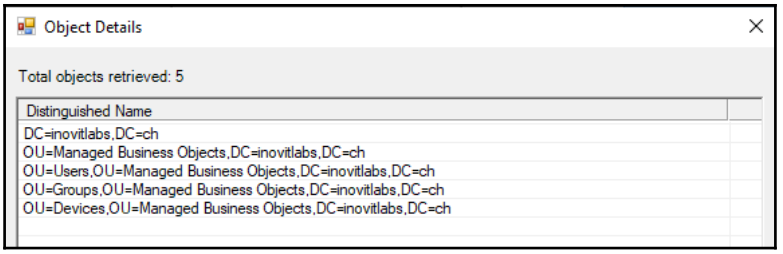
Full Import profile run

3. The sync engine imports the organizational structure (placeholder) into the CS:



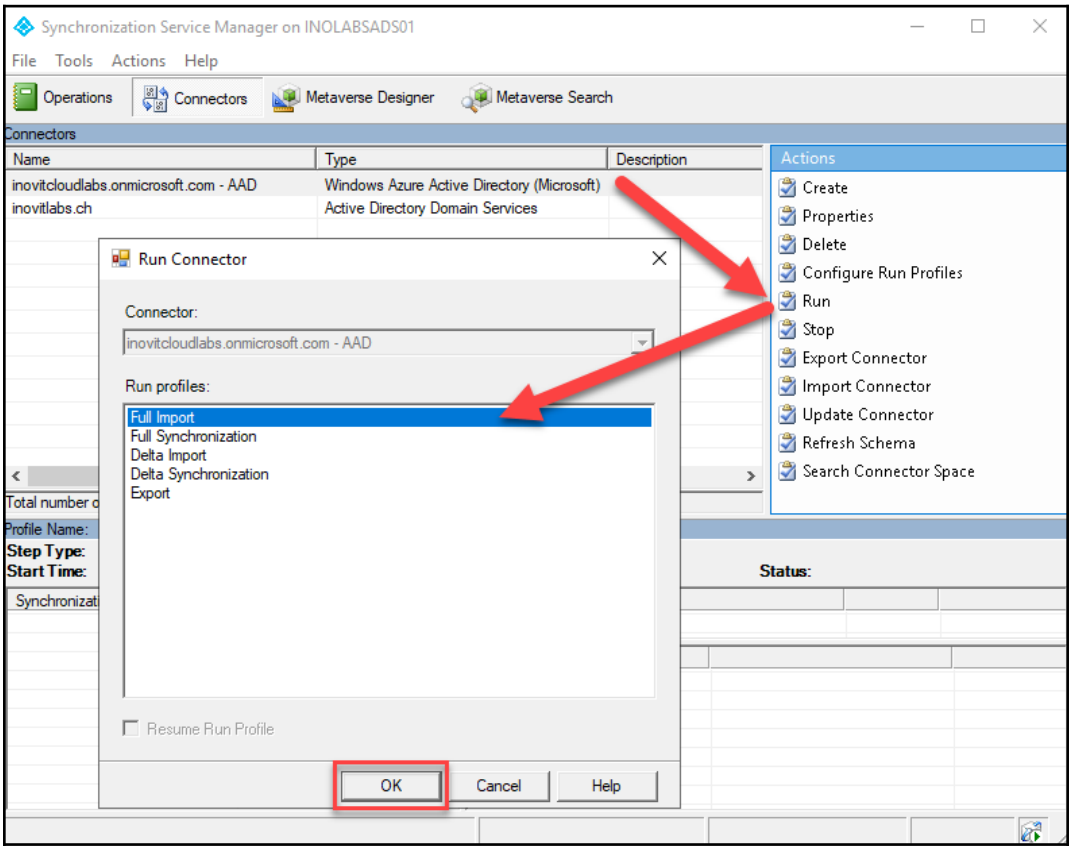
Statistics after Full Import

4. You see the imported organizational structure:



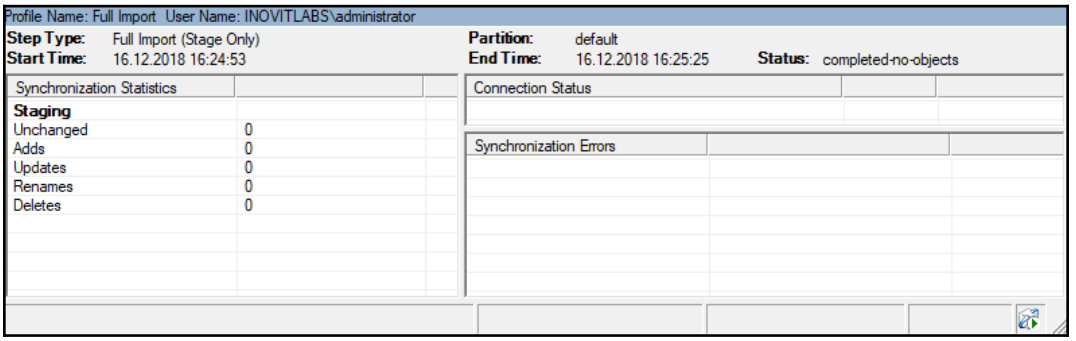
Imported objects

5. Run a **Full Import** on the connected **Azure Active Directory**:



Full Import on Azure AD

6. No objects are created in the CS. Remember that we already have users in the Azure AD—there's no user write-back available:



Full Import Azure AD statistics

Also, there's no interaction with the MV with an import flow. In the next section, you'll learn more about placeholder objects.

Placeholder objects

An example of a placeholder object is the containers to validate the path to the AD objects. We can also talk about a placeholder object to which the manager attribute refers. As a rule, we can say that placeholder objects don't contain values and are not linked with the MV.

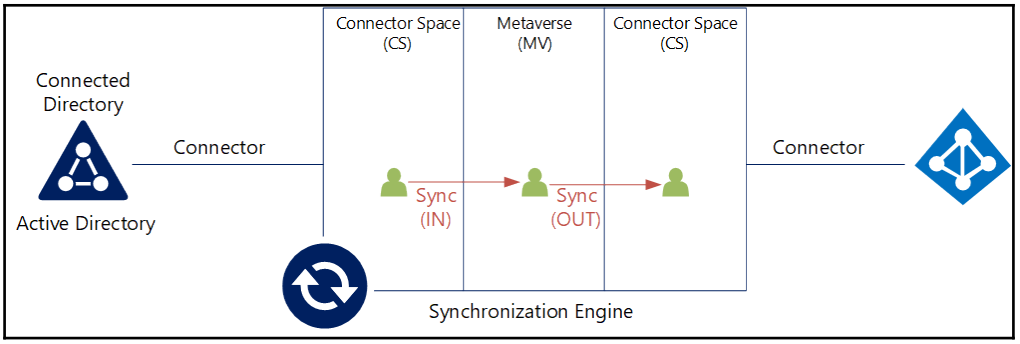
Run profiles—one per forest, one step per domain:

- We import (stage) all objects into the CS (even filtered by the Connector Filter)
- Full Import—all objects
- Delta Import—only changed objects

Next, we'll move to the synchronization flows.

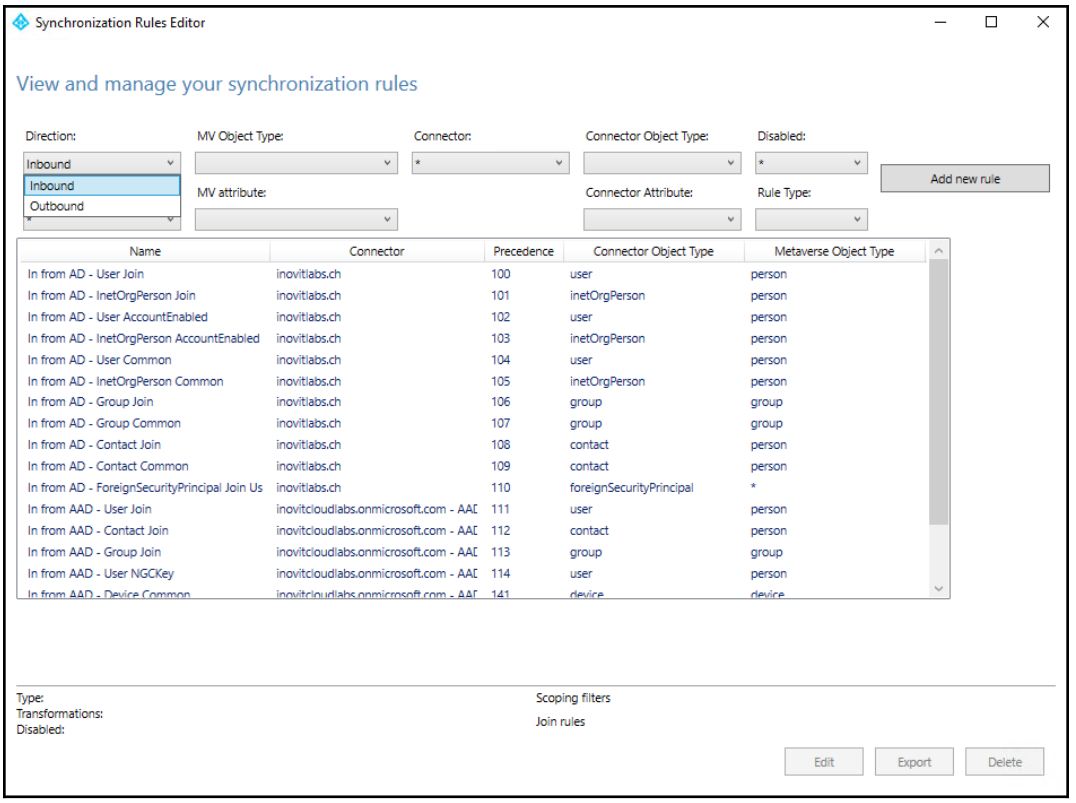
Synchronization flows

Synchronization flows are specific to the processes between the CS and the MV objects. As a rule, we can say that multiple CS objects can be linked to one MV object. This is typical because the MV represents the unique identity object over all connected systems. On the other hand, a CS object can't have a link to multiple MV objects, as shown in the image below:



Synchronization flow overview

The synchronization flows are configured over the synchronization rules that you can view in the Azure AD Connect **Synchronization Rules Editor**:

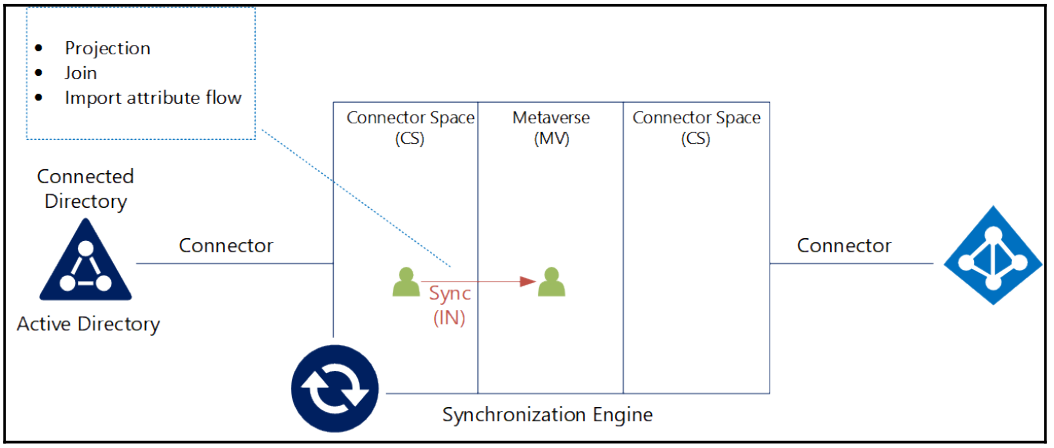


Synchronization rules editor overview

In `Chapter 3`, *Exploring Advanced Synchronization Concepts*, we'll discuss the synchronization rules in detail. For now, we'll focus on the main functions and explanations.

Inbound synchronization

The inbound synchronization is defined by a subset of synchronization rules and contains the terms shown in the following diagram. If we use an inbound synchronization, the synchronization engine will update the changes between the MV and the CS. The inbound synchronization happens when the content of the MV is updated by using the data in the CS:

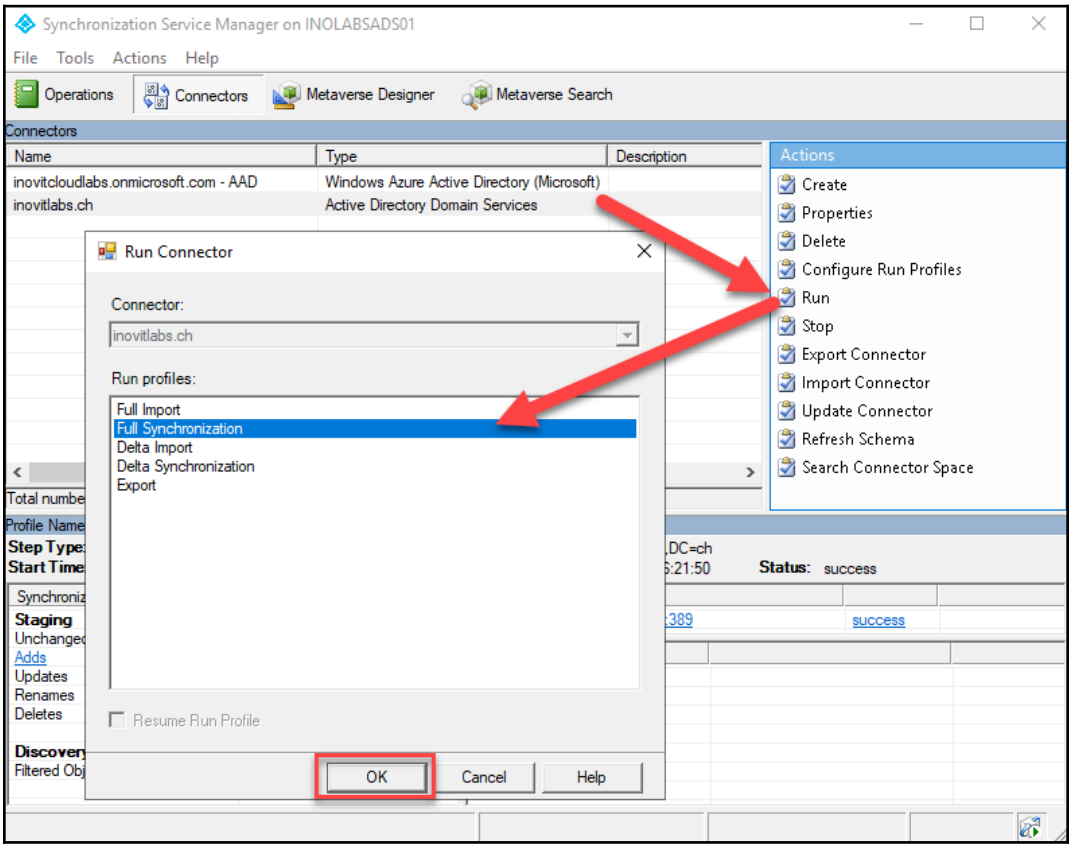


Inbound synchronization overview

Another term for the provisioning is **Projection** and it's part of the outbound synchronization. We can also say that provisioning is an object-level operation because the synchronization engine creates a new object in the MV based on the object in the CS and creates a link between them. The join is in the special process, which creates the link. So, the join is also an object-level operation. The import attribute flow is used from the synchronization engine to update attribute values of the MV object. We can say that the import flow is attribute-level-based and requires a link between the CS and the MV object.

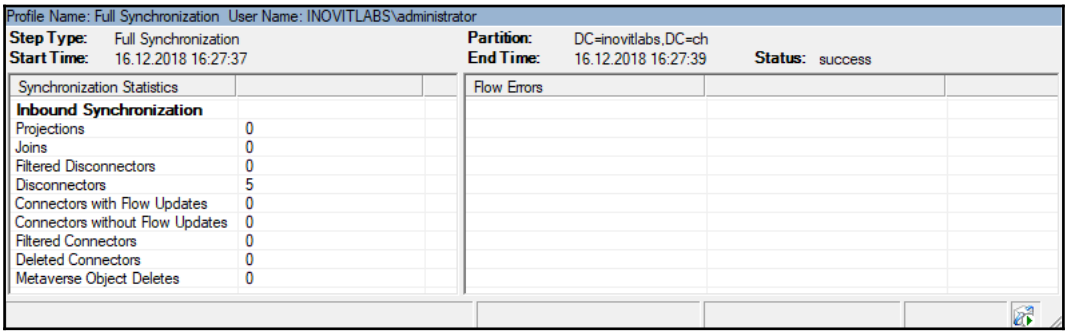
Let's take a look at our Azure AD Connect:

1. Log into your Domain Controller and open the Azure AD Connect **Synchronization Service Manager**.
2. Run a **Full Synchronization** on the local AD:



Full Synchronization run on the local AD

3. Nothing happens on the MV. You will only find **5 Disconnectors** for the organizational structure (placeholders):

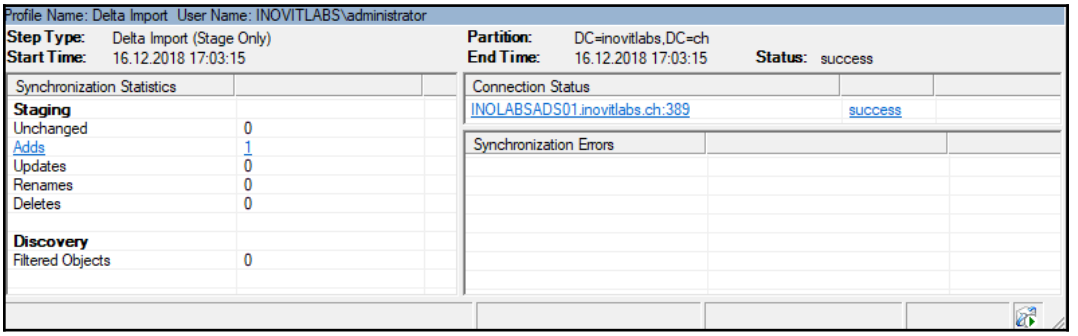


Full Synchronization statistics

Let's create a test user with the same `UserPrincipalName` as an existing user in Azure AD. In my case, I use `jochen.nickel@inovitlabs.ch`:

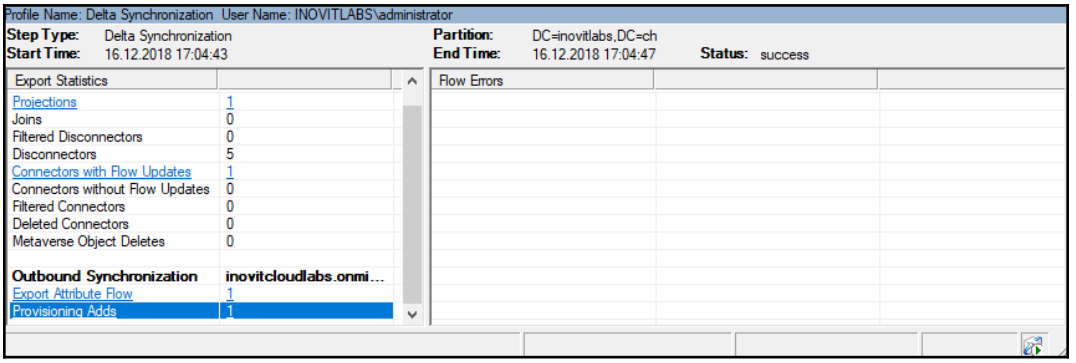
```
New-ADUser -Name "Jochen Nickel" -GivenName Jochen -Surname
Nickel -SamAccountName jochen.nickel -UserPrincipalName
jochen.nickel@inovitlabs.ch -path "OU=Users,OU=Managed
Business Objects,DC=inovitlabs,DC=ch" -AccountPassword
(ConvertTo-SecureString "Pass@word1" -AsPlainText -Force) -
Enabled $True
```

4. Run a **Delta import** on the local AD:



Delta Import statistics on local AD

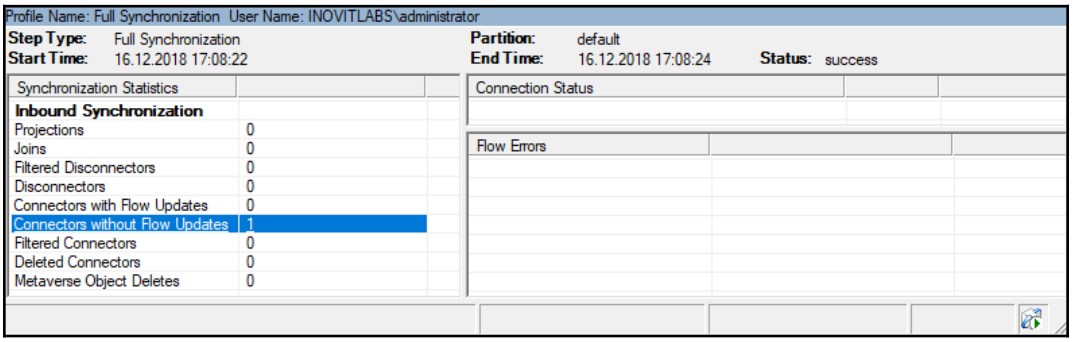
5. Run a **Delta Synchronization** on the local AD:



Delta synchronization on local AD statistics

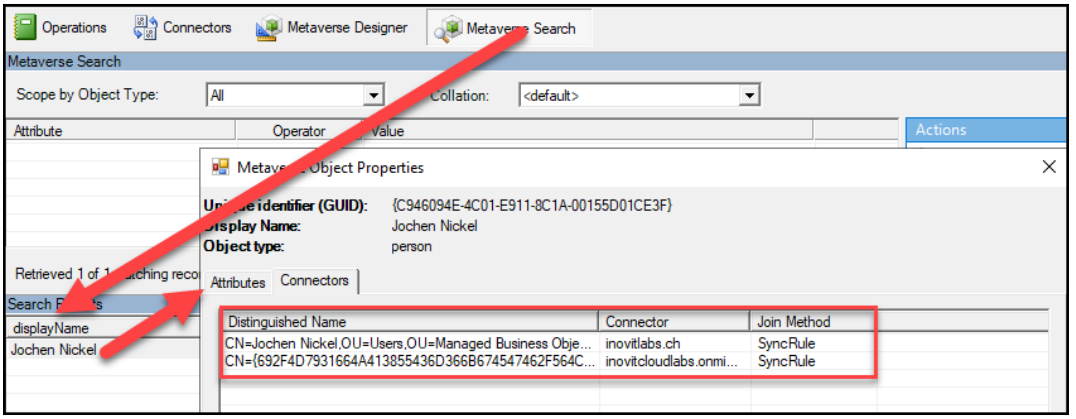
In the actual state, we would have a provisioning of the user in the Azure AD. For this reason, we need to run a **Full Synchronization** on the Azure AD to get all the changes. This procedure avoids the making of a wrong provisioning, because we want to join the user based on the `UserPrincipalName`.

6. Run a **Full Synchronization** on the Azure AD:



Full synchronization statistics

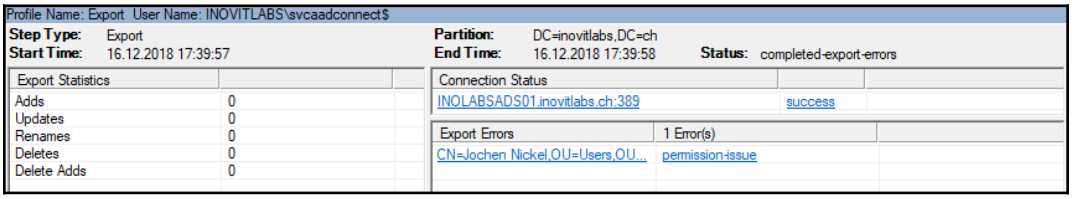
7. Click on **Connectors without Flow Updates** and inspect the changes.
8. Run an **Export** on the Azure AD which will recognize that the user will be joined by `UserPrincipalName`. We prove this with a **Metaverse Search**:



Metaverse Search options

You synchronized your first user, including a join operation! You can follow up by mapping all the accounts. This is how you can map existing on-premises accounts with Azure AD accounts.

If you run an **Export** on the AD, you'll receive a **permission issue** error:



Export statistics on AD

Modify the cmdlets to your environment and execute them to grant the AD Management Account the permissions to write to `ms-ds-consistencyGuid`:

```
$accountName = "INOVITLABS\svcaadcadma"
$ForestDN = "OU=Managed Business Objects,DC=inovitlabs,DC=ch"
$cmd = "dsacls '$ForestDN' /I:S /G '`"$accountName`":WP;ms-ds-
consistencyGuid;user'"
Invoke-Expression $cmd
```

Run the **Export** on the AD again and the error will disappear.

You can also use the email address or write `ImmutableId` to a PowerShell script. You can use the following script for this purpose, `https://gallery.technet.microsoft.com/scriptcenter/Convert-between-Immutable-e1e96aa9` or the following procedure:

```
$guid = (Get-ADUser -Identity jochen.nickel).ObjectGUID
$immutableid=[System.Convert]::ToBase64String($guid.tobytearray())
Set-MsolUser -UserPrincipalName jochen.nickel@inovitlabs.ch -
ImmutableId $immutableid
```

After you successfully join all the users, you can run a complete default sync cycle with the following PowerShell cmdlet:

```
Start-ADSyncSyncCycle -PolicyType Initial
```

This would happen if you enable the synchronization in the installation and configuration wizard of Azure AD Connect.

By default, the Azure AD Connect Sync Engine schedulers runs the following on the initial run:

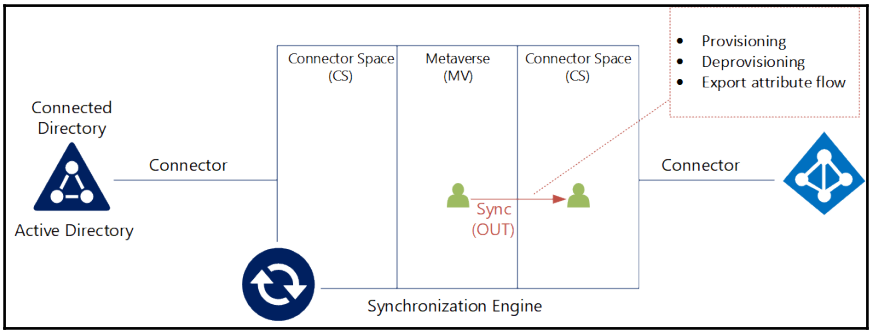
- AD—Full Import
- AAD—Full Import
- AD—Full Sync
- AAD—Full Sync
- AAD—Export
- AD—Export

Now that we've done the practical tasks, let's look at the rest of the theoretical information.

Outbound synchronization

With outbound synchronization, you update the exported objects when an MV object is changed but not deleted. The primary purpose of the outbound synchronization is to check the changes to MV objects, which need to be updated in the CS.

With outbound synchronization, we use the following three processes. You will get a provisioning process if changes are applied to an object in the MV. The deprovisioning process always depends on the provisioning itself, because the provisioning process is the only one that can initiate the export attribute flow. The objects in the CS during an export are provided with the pending export flag to make them export objects. You can see the process in the following diagram:

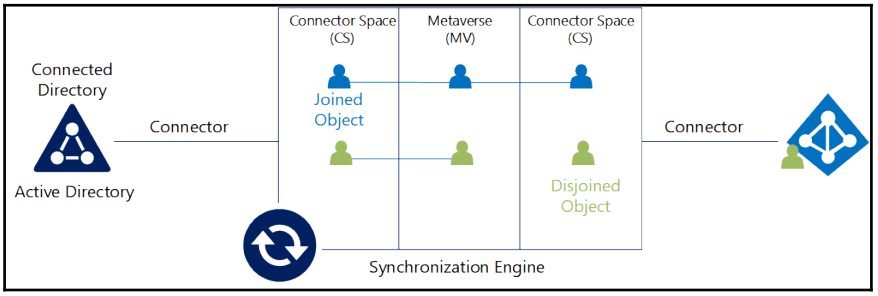


Outbound synchronization overview

If changes happen to an MV object, the synchronization engine can create joined objects as part of the provisioning process. Furthermore, the synchronization engine can rename a joined object and disjoin an object.

Joins

Joins are an excellent functionality. We call a linked object in the CS, with an object in the MV a joined object. Otherwise, it's a disjoined object. The following diagram shows the Joins concept:



Joins overview

With the synchronization process, an object from the CS will be a joined object. In this case, the related attributes can flow. The attribute flow is in both directions.

The disjoined object usage delivers such that the required information about the object is already stored and they are convertible in both ways. With an import you always create a disjoined object as well with the export.

Connector objects

Now that we have discussed the join operations we will explain the different connector objects, which are used by the system to build the connection between the connector space and the metaverse object.

There are two types of connector objects:

- Connectors
- Explicit connectors

A connector is an object in the CS. The object is linked to an object in the MV, so every rule applies to this object. Furthermore, an explicit connector exists. With this type of connector, the object is linked to an object in the MV. This type of connector can only be created with the Joiner functionality, which isn't available in Azure AD Connect.

Disconnector objects

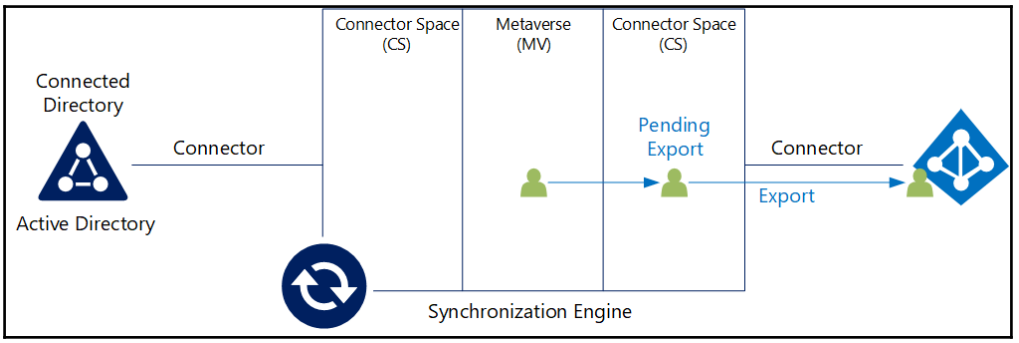
There are three types of disconnector objects:

- Disconnectors
- Explicit disconnectors
- Filtered disconnectors

We speak about a disconnector when an object in the CS isn't linked to an object in the MV. An explicit disconnector is also not a linked object in the MV and can only be joined with the Joiner toolset of the synchronization engine. This feature is not available in Azure AD Connect. The meaning of a filtered disconnector is an object in the CS, that is prevented from being joined or projected to an object in the MV.

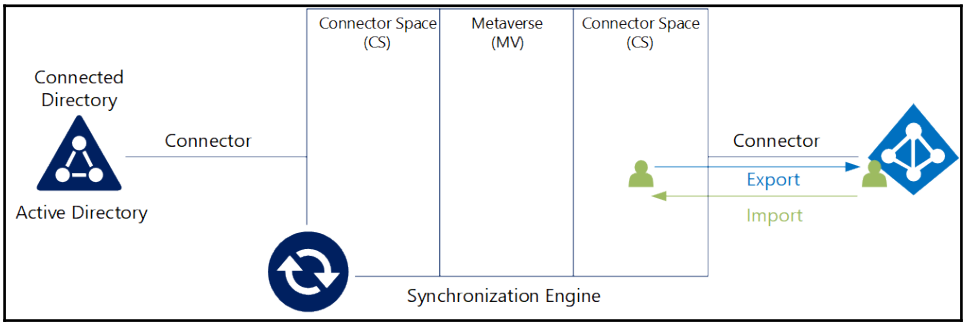
Export flow

During the export process, the synchronization engine checks all the exported objects with the **Pending Export** flag in the CS, and an update will be sent to the data source:



Export flow overview

At this moment, the synchronization engine can only validate whether the export was successful. It needs an import to prove that the exported values are correctly transported to the repository:



Import flow to prove the export

Now that you are equipped with the essential information about Azure AD Connect and synchronization tasks, we'll jump into the details of the synchronization rules editor in `Chapter 3`, *Exploring Advanced Synchronization Concepts*.

Summary

In this chapter, we discussed the most important identity synchronization tools: Microsoft Identity Manager and Azure Active Directory Connect. We walked through the typical synchronization scenarios. Now you're able to adopt the best scenario for your requirements. In the *Synchronization terms and processes* section, we took a deep dive into the synchronization service, so you know exactly what's happening under the hood, which will help you to avoid mistakes and provide better troubleshooting for synchronization errors.

In the next chapter, we'll explore additional filtering, join attributes, declarative provisioning options, and generic connector usage.

3
Exploring Advanced Synchronization Concepts

In current projects and workshops, we often see that administrators or consultants find handling the Azure AD Connect synchronization engine difficult, especially if the out-of-the-box functionality configured by the wizard doesn't work as expected in their scenarios. The chapter will give you all the needed knowledge so that you can handle basic and advanced synchronization requirements.

In this chapter, we'll discuss advanced synchronization concepts. We'll look into the synchronization rules and work with practical situations to provide you with the information and practical experience you need to solve common requests in your projects or solutions. Furthermore, we explain the declarative provisioning and expressions concept and use them directly in real-world examples. Another topic in this chapter is the connection to an additional untrusted Active Directory forest. Finally, we'll give you an idea of the special considerations you need to handle.

This chapter is organized into the following topics:

- Understanding declarative provisioning and expressions
- Synchronization rules explained
- Special considerations in advanced synchronization concepts

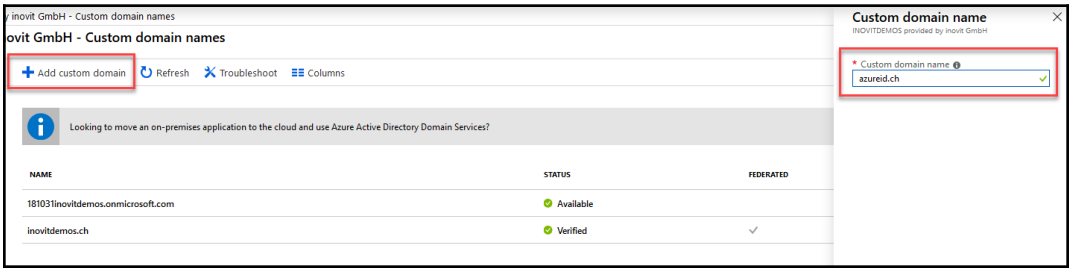
We will start with the steps to prepare our lab environment.

Preparing your lab environment

To work through the guidance provided in this chapter, we need to arrange some preparation tasks. You need to provide an additional public DNS suffix (which in my case is `azureid.ch`) that represents `YOURDOMAIN2.COM`. We need to add this other domain as a custom domain in the first Azure AD tenant (`YOURDOMAIN1.ONMICROSOFT.COM`), which we used in `Chapter 2`, *Understanding Identity Synchronization:*

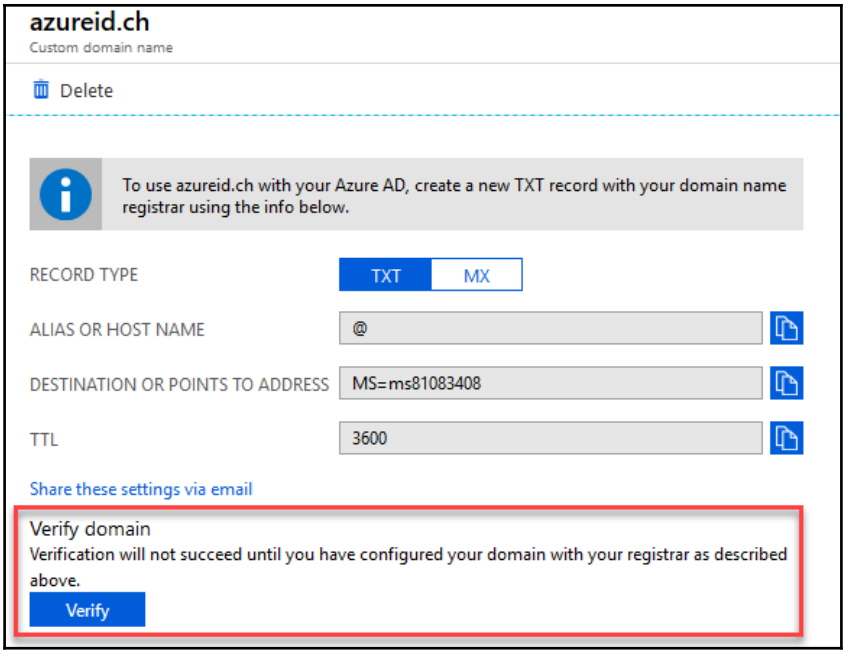
Use the following steps to start the configuration:

1. Open the Azure Portal: `https://portal.azure.com`.
2. Navigate to the Azure AD blade.
3. Click **Custom domains.**
4. Click **Add custom domain.**
5. Use your additional domain name:



Adding a custom domain

6. Configure your public DNS to represent the following verification entry:

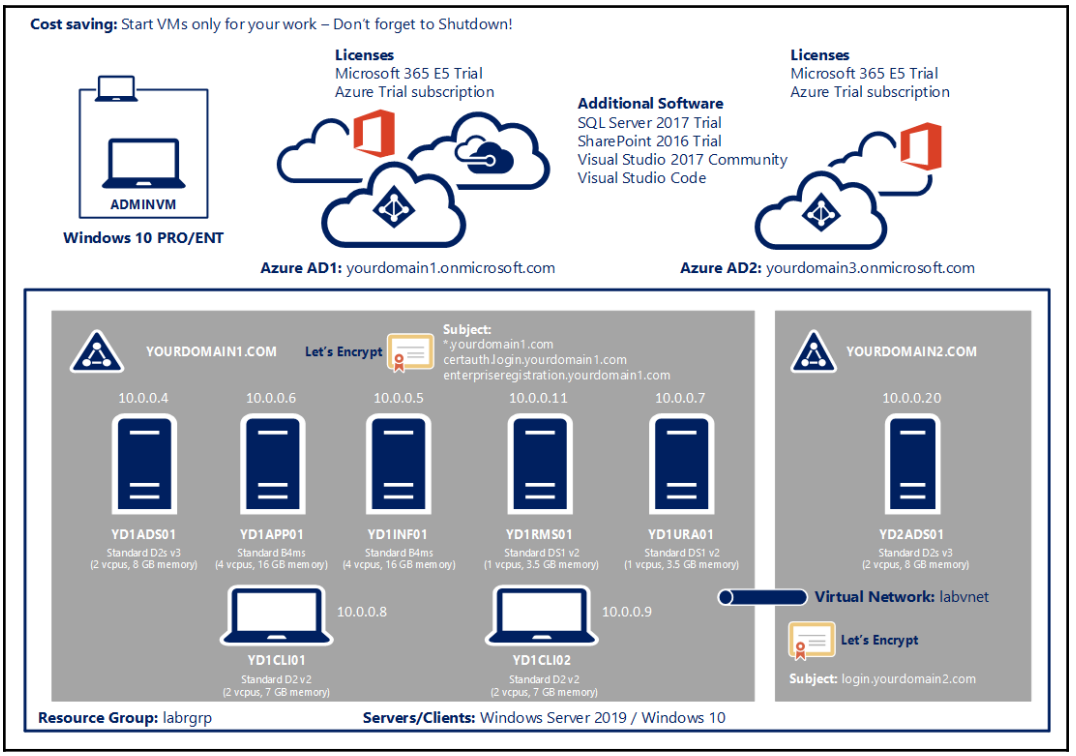


Custom domain verification

7. Click **Verify**.
8. The following result is expected:

Verified domains overview

The following diagram shows the complete lab environment we'll use in this book:



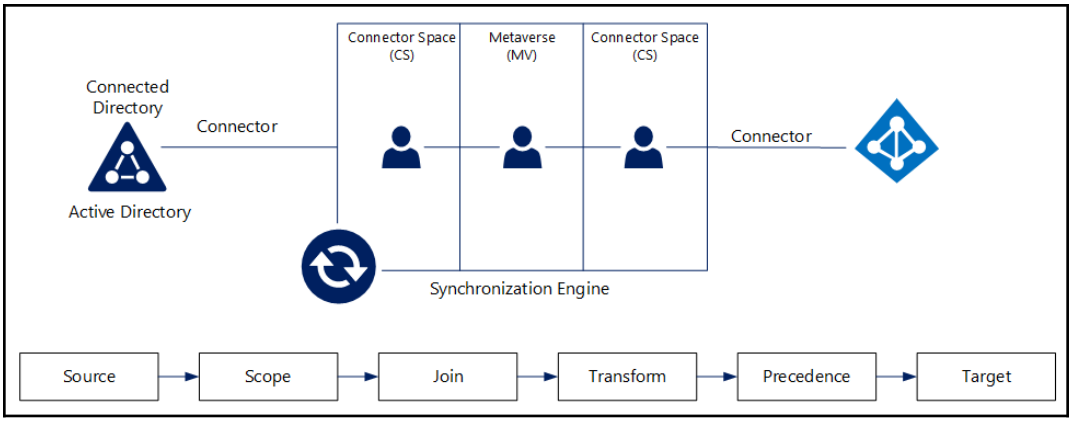
Lab environment overview

Furthermore, we need to configure a new Active Directory Forest that uses the new domain name. Use the preceding specifications to create the virtual machine to host the domain controller. Create the same organizational structure and users, just like you did for the first Active Directory forest that we used in the `Chapter 2`, *Understanding Identity Synchronization*. You'll find all the scripts in this book's code package. Don't worry—we'll do the configuration in this chapter to avoid errors in your lab. After preparing the lab environment, we can start working through this chapter.

We are focusing on the synchronization process in this chapter to adopt the authentication parts you can use the resources from `Chapter 6`, *Managing Authentication Protocols*, `Chapter 7`, *Deploying Solutions on Azure AD and ADFS*, and `Chapter 8`, *Using the Azure AD App Proxy and the Web Application Proxy*.

Understanding declarative provisioning and expressions

The easiest way to explain declarative provisioning is as follows: objects are processed from the source connected directory to the target source by evaluating how the objects and the associated attributes should be transformed. This is controlled by inbound rules from the connector space to the metaverse and outbound rules from the metaverse to the connector space. The following diagram gives you an overview of all of the components:



Declarative provisioning options and components overview

Declarative provisioning provides the following capabilities:

- The only way to configure the sync engine
- Functions to configure attribute flows
- Precedence is on SRs (not on Connectors)
- **MV—**deletion rules now use declarative provisioning
- Introduces parameters, such as `%Domain.Netbios%`
- Configured through PowerShell

The attribute-flow expression language can be explained as follows:

- Written in **Visual Basic for Applications** (**VBA**)
- Stricter syntax:
    - Useful errors for easy troubleshooting
    - Strongly-typed for different data types

- Evolved expressions:
    - `[attributename]`, `%parametername%`
    - `&H` (hexadecimal value)
    - Constants—`CRLF`, `True`, `False`, `NULL`

It brings the following attribute flow operators into the game:

- **String concatenate**: `&`
- **Mathematics**: `+`, `-`, `*`, `/`
- **Comparison**: `=`, `<`, `>`, `<>`, `<=`, `>=`
- **Evaluation order**: `( )`
- **Logical operator**: `&&` (and) `||` (or)

It also provides a lot of functions for the attribute flow, as follows:

Conversion	`CBool`
	`CDate`
	`CGuid`
	`ConvertFromBase64`
	`ConvertToBase64`
	`CNum`
	`CRef`
	`CStr`
	`StringFromGuid`
	`StringFromSid`
		Math	`BitAnd`
			`BitOr`
			`RandomNum`

Date/time	`DateAdd`
	`DateFromNum`
	`FormatDateTime`
	`Now`
	`NumFromDate`
		Multi-valued	`Contains`
			`Count`
			`Item`
			`Join`
			`RemoveDuplicates`
			`Split`

Directory	`DNComponent`
	`DNComponentRev`
	`EscapeDNComponent`
		Program flow	`Error`
			`IIF`
			`Switch`

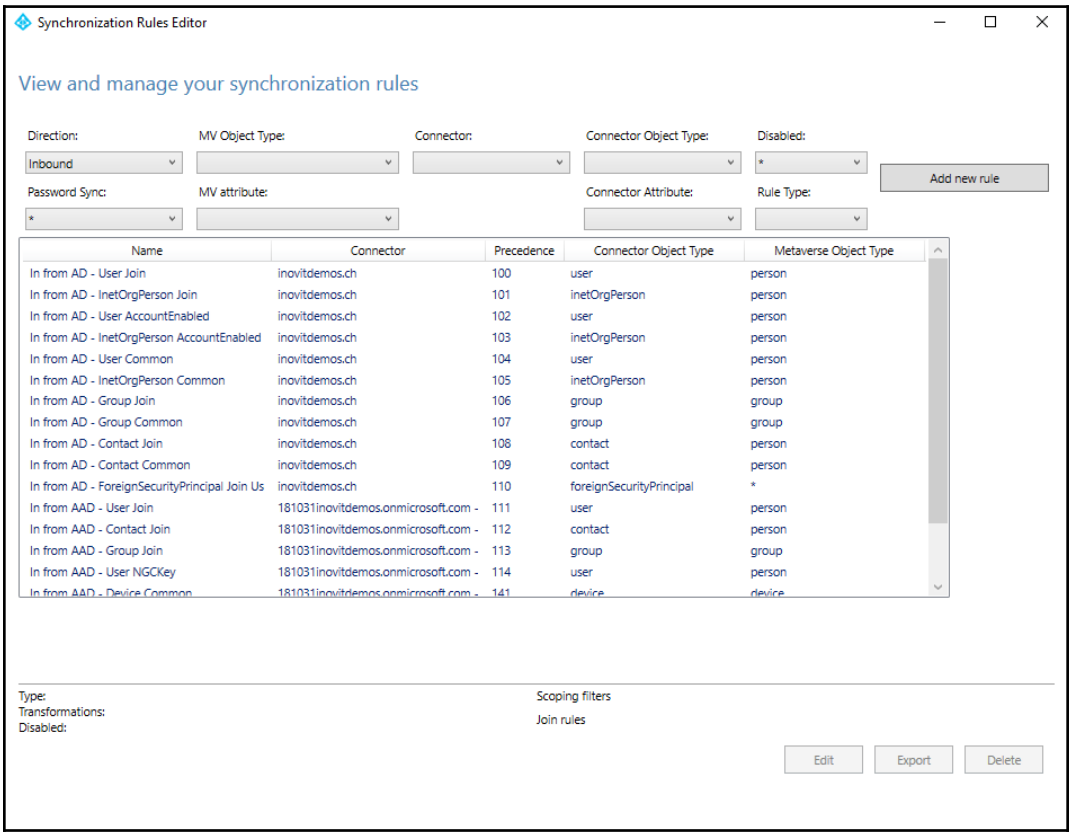
Inspection	`IsBitSet`
	`IsDate`
	`IsEmpty`
	`IsGuid`
	`IsNull`
	`IsNullOrEmpty`
	`IsNumeric`
	`IsPresent`
	`IsString`
		Text	`GUID`
			`InStr`
			`InStrRev`
			`LCase`
			`Left`
			`Len`
			`LTrim`
			`Mid`
			`PadLeft`
			`PadRight`
			`PCase`
			`Replace`
			`ReplaceChars`
			`Right`
			`RTrim`
			`Trim`
			`UCase`
			`Word`

In the following section, we'll explain these components in detail to provide you with a better understanding.

Synchronization rules explained

Azure AD Connect uses an extra user interface in the **Synchronization Rules Editor** to manage the synchronization logic. In the following screenshot, you can see all of the synchronization rules have been created for your basic configuration. Every entry is one synchronization rule. In the **Direction** dropdown, you can choose between two different types: **Inbound** and **Outbound**. Practically, we say that the **Inbound** and **Outbound** synchronization is always viewed from the metaverse perspective. In my explanations, I'll use the **inbound synchronization rules** because we will find the related information there.

In the following screenshot, we can see the connected Active Directory forest (`inovitdemos.ch`) and that it doesn't have any services, such as Exchange or Skype for Business, and no synchronization rules have been created for these services:

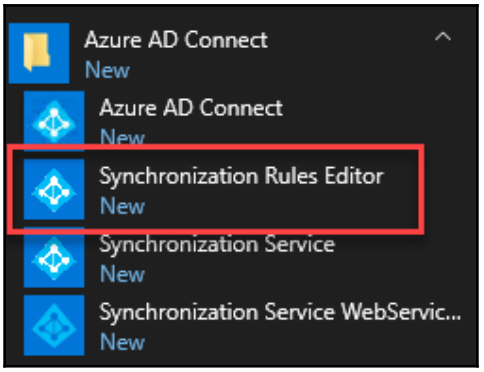


Synchronization rules overview—Inbound

With the following steps, we will gather more information on the practical usage of the tool:

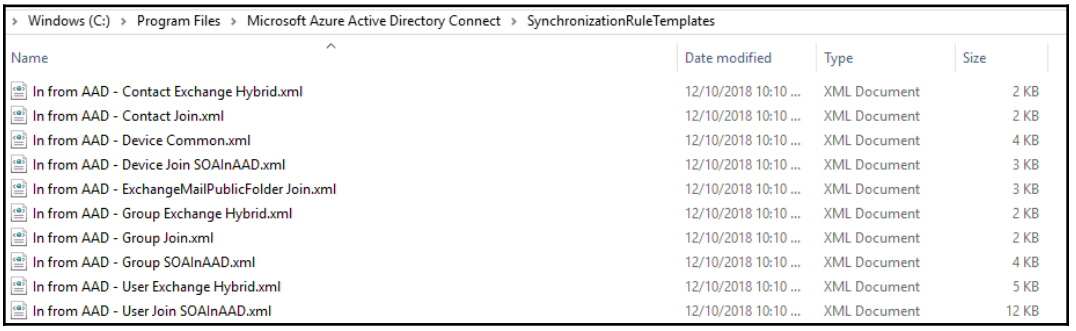
1. View this configuration on your **YD1ADS01**.
2. Click on **Start** and navigate to **Azure AD Connect**.

3. Click on **Synchronization Rules Editor**:



Start Menu—Synchronization Rules Editor

AAD Connect provides several default synchronization rules based on your configuration. The templates for these rules can be found under `C:\Program Files\Microsoft Azure AD Connect\SynchronizationRulesTemplates` and are XML-based:

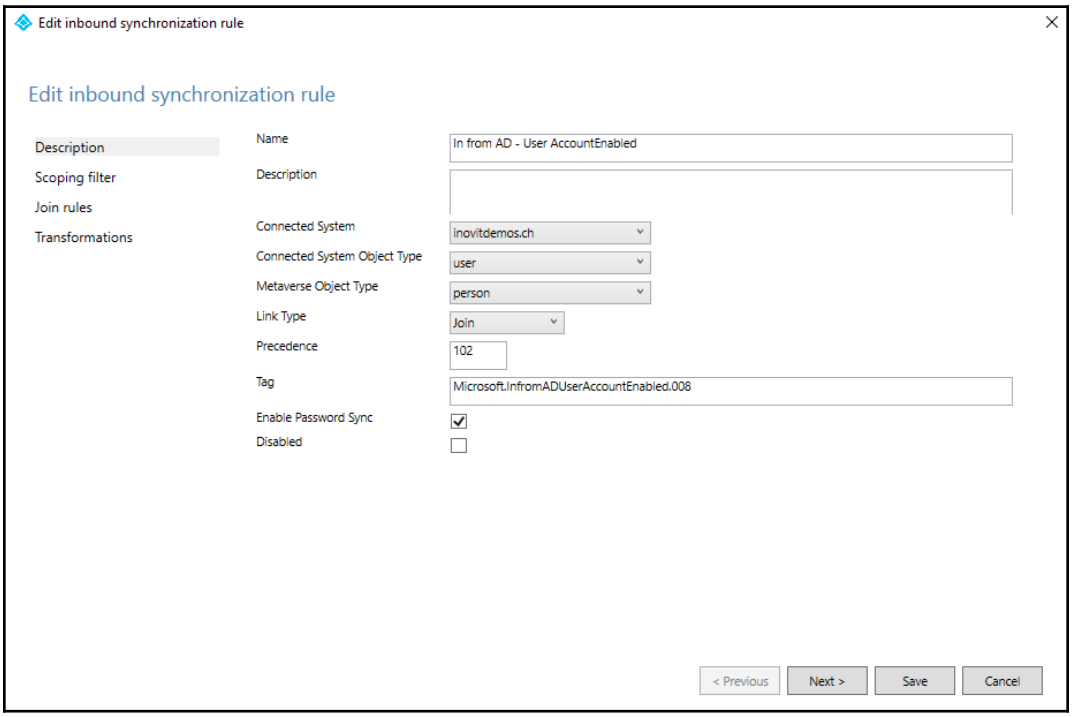


Synchronization rules templates overview

The screenshot of the **Synchronization Rules Editor** shows the different rules and their precedence.

Microsoft always generates the first rule with the number `100` and iterates over all connectors. They read the first item in that file (sorted by precedence). It has the name of a rule and the criteria for being added. Then, they generate the first rule with the number `100` and iterate over all connectors. If there are multiple connectors, the `whenCreated` date/time is used as the arbitrator.

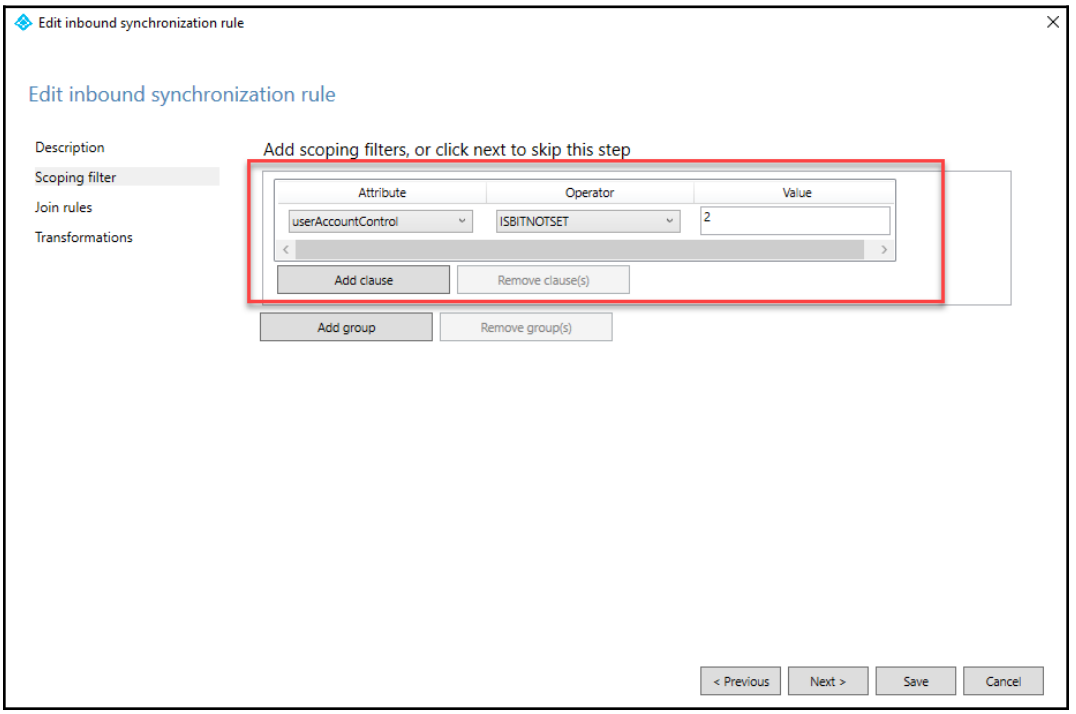
In the following screenshot, we'll take an in-depth look into a **synchronization rule**:



Inbound synchronization rule details

Here, we'll find detailed information about the **Connected System** for which the rule applies. Further details are the object type in the metaverse and the connector space of the synchronization engine. As we can see in the preceding example, the object type in the connector space is called a user, which represents a user account in an Active Directory, for example. The **Metaverse Object Type** for a user is always a **person**. Under the **Link Type**, you can choose between **Join**, **StickyJoin**, and **Provision**. We already discussed the join and provision option in `Chapter 2`, *Understanding Identity Synchronization*. The **StickyJoin** option can be used if the provisioning of a new object to the metaverse is prohibited. Furthermore, the rule is enabled for password synchronization.

In the following screenshot, we explore the **Scoping filter** capabilities. The following rule applies only if the user is enabled in the Active Directory. We use the **userAccountControl** attribute to identify the status of the user. This screenshot shows us that the value isn't required to have the bit `2` not set. The value for an enabled user in Active Directory is `512`, and `514` for a disabled one:

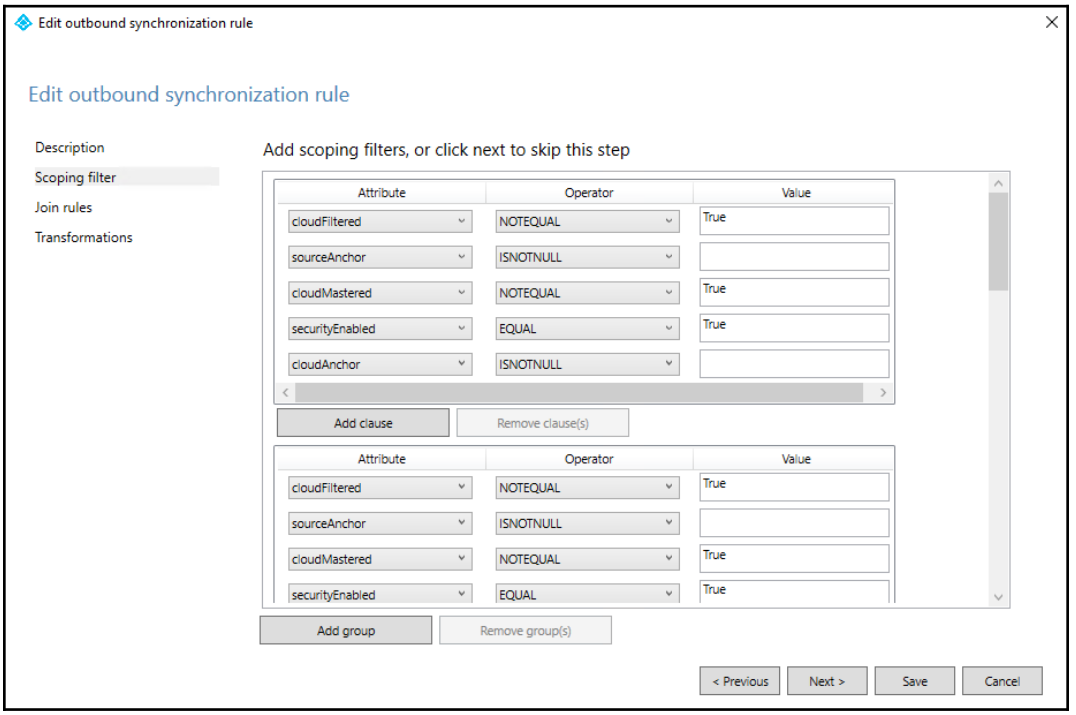


Scoping filter options

Within the **Scoping filter**, we can see that there are groups and clauses that can be nested. To apply a rule, all clauses inside a group need to be met. A rule has two types of logical operators: the logical OR operator that's used between groups and the logical AND within a group.

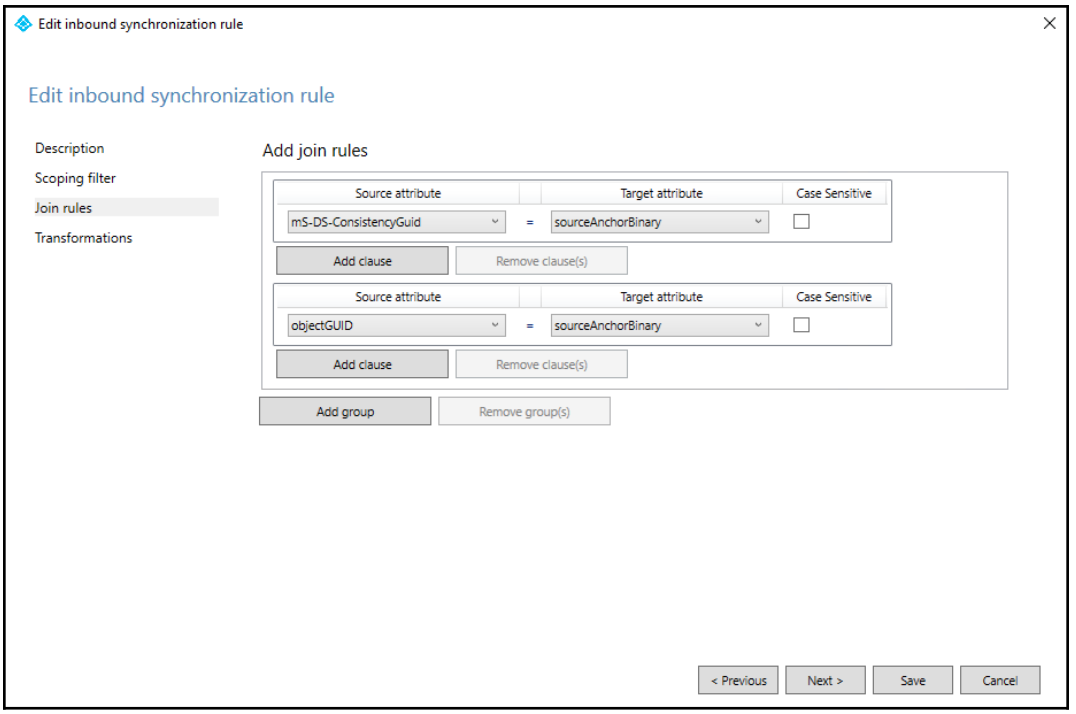
We'll take a closer look at the `Out to AAD—Group Join` rule, which is shown in the following screenshot. We can find an example of the different group types and how they get handled.

The rule shown in the following screenshot regulates how and which groups are provisioned to the Azure AD. Also, other attributes will be checked from the rule:



More Scoping filter options

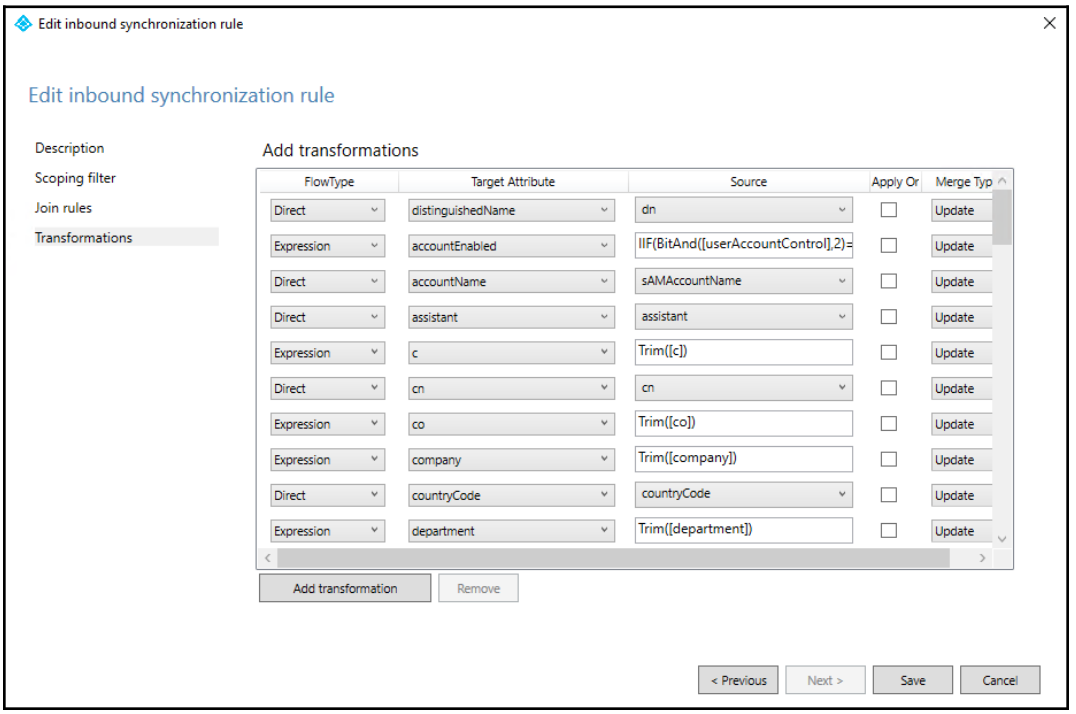
In the following example, we look into the `In from AD—User Join rule`, where we find **ms-DS-ConsistencyGuid** and **objectGuid** handling for joining users from the connector space to the metaverse. Keep in mind that join rules will only be checked once. The link between the connector space and the metaverse object exists until the conditions in the synchronization rule will not be achieved. It's important to know that the synchronization engine will show an error when multiple join rules want to be applied:



Join rules example

For transformations types, such as constant, direct or expression can be used. If we use a constant flow, we can write a value directly into the chosen attribute. With a direct flow, the attribute from the source system will be used in the target attribute. With **Visual Basic for Applications** (**VBA**), we can develop extended configurations. The syntax for attributes is the square brackets, such as **company** or **department**, as shown in the following screenshot. Keep in mind that you need to work with case-sensitive functions and attribute names. You can also use nesting options in your rules.

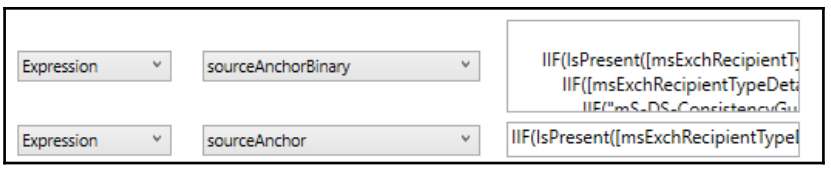
The following screenshot shows the transformations of the `In from AD—User common` rule:



Transformation rule example

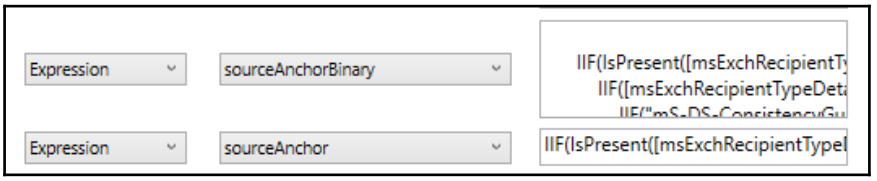
We worked through individual synchronization rules. Sometimes, many synchronization rules are applied to one target attribute. The `sourceAnchor` in the following screenshot can be used as an example. There, you can see that two rules have been applied to the attribute. Now, precedence comes into the game and decides which rule will be used. View the following screenshots to follow the explanation:

- `In from AD—User AccountEnabled`:



Expression example for AccountEnabled

- `In from AD—User Common`:



Expression example sourceAnchor

In the case of `sourceAnchor`, an enabled account from the connected Active Directory will be populated to the Azure AD. If no enabled account is available in any connected Active Directory, the `In from AD—User Common` rule applies. The default behavior for multiple connected Active Directories is that the `In from AD—User Join` rule has the highest precedence. So, the system iterates through all connected Active Directories.

Now that we have introduced you to the concepts of declarative provisioning and synchronization rules, we'll configure some examples.

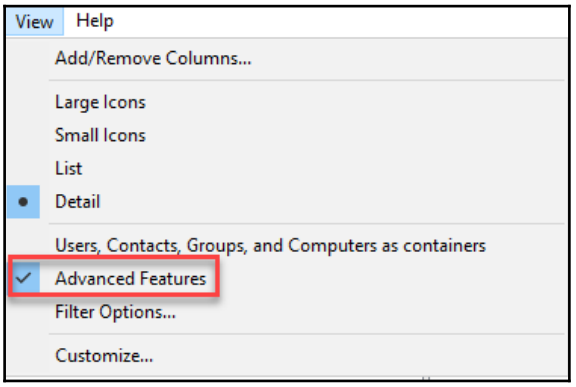
Special considerations in advanced synchronization concepts

In this section, we'll start using our knowledge in practical examples. First, we'll explore some essential functions that can be used out of the box. In some environments, you have the requirement that an organization has an **organizational unit** (**OU**) filter in place, where all users are included in this OU. But now that you need to filter out, this shouldn't be synchronized to the Azure AD. Furthermore, we'll integrate a second AD forest and use PowerShell to configure the synchronization rules.

Using standard filters to exclude users and groups

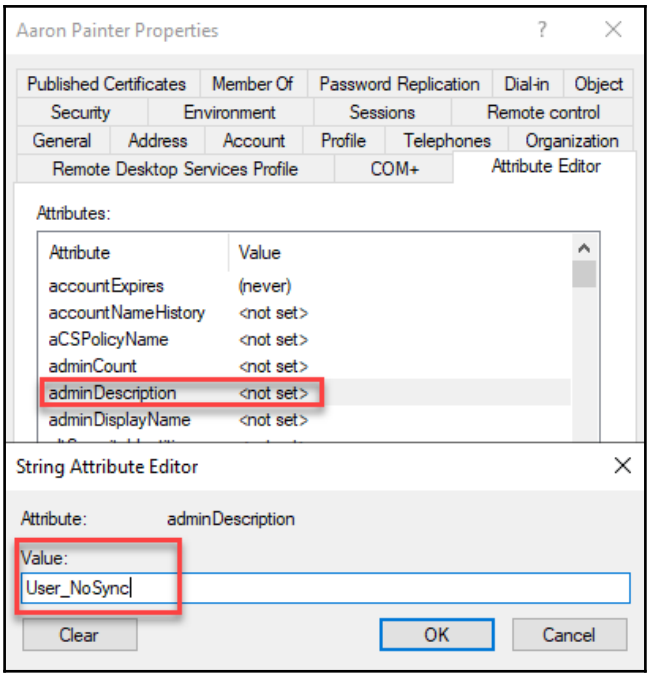
In this section, we'll use the standard filtering options to exclude users and groups to be synchronized to the metaverse:

1. Log in as domain administrator to your **YD1ADS01**.
2. Open the Active Directory Users and Computers console (`dsa.msc`).
3. Be sure that you are in the advanced features view:



Active Directory Users and Computers—Advanced Features option

4. Choose one of your users and move to the **Attribute Editor** tab.
5. Edit the **adminDescription** attribute and enter `User_NoSync`, where `User_` is the important part:

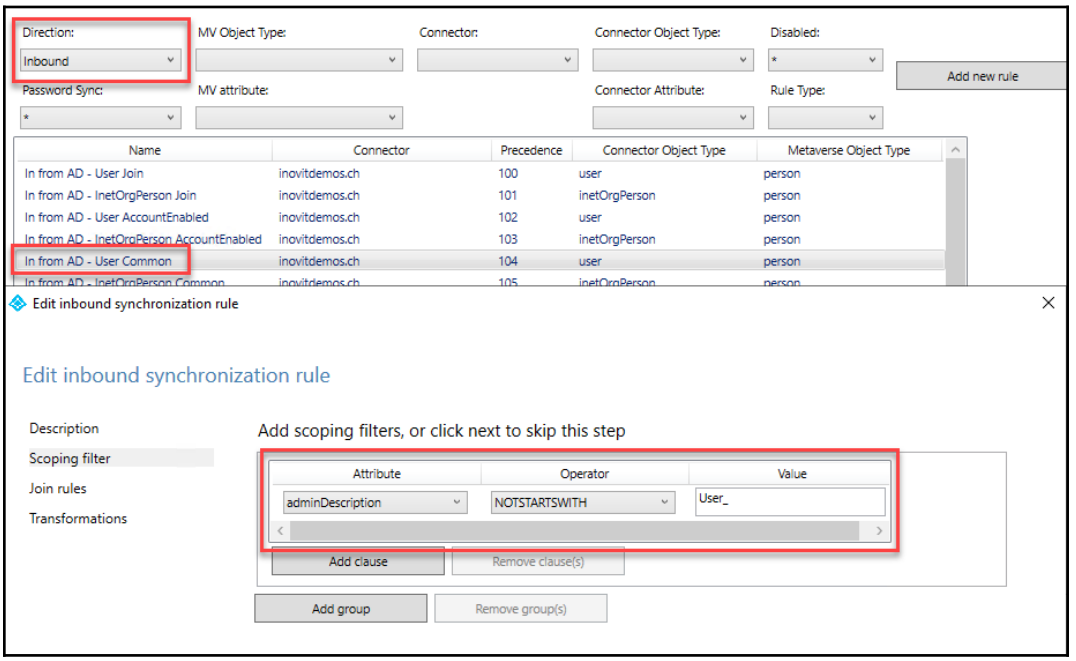


adminDescription filtering option

6. Save your settings and minimize the console.
7. Open the **Synchronization Rules Editor**.
8. Edit the rule `In from AD - User Common`.
9. Click on **Scoping filter**.
10. You'll see a **Scoping filter** with the following clause:

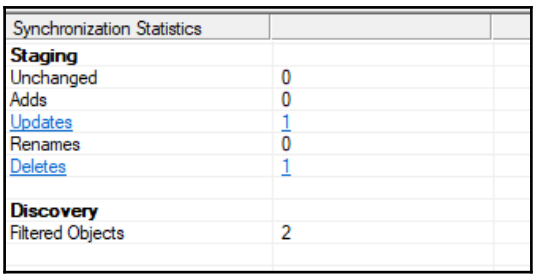
```
adminDescription NOTSTARTSWITH User_
```

The following screenshot shows the configuration to filter users to synchronize to Azure AD by filling the `adminDescription` attribute:



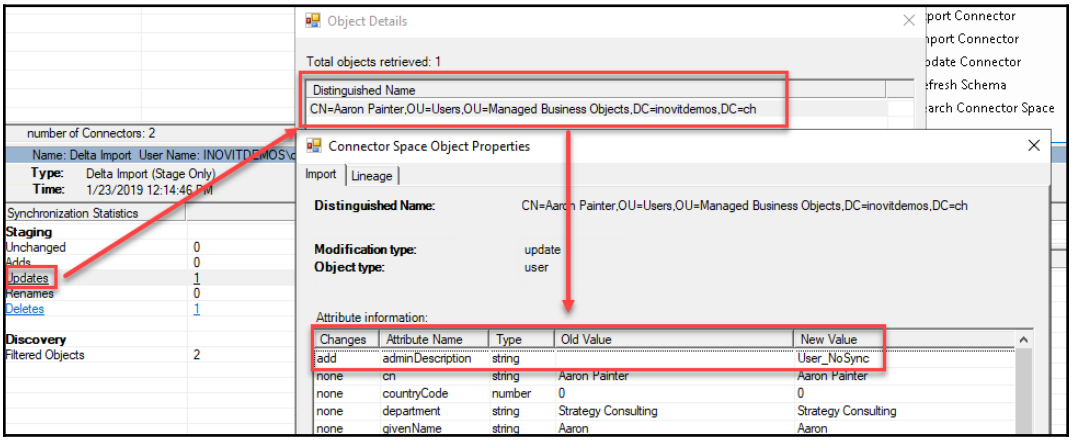
Rule configuration to filter specific users

11. Click **Cancel** and minimize the synchronization rules editor.
12. Open the **Synchronization Service Manager**.
13. Click on **Connectors** and start a delta import on your Active Directory connector.
14. You'll get an update in the sync statistics:



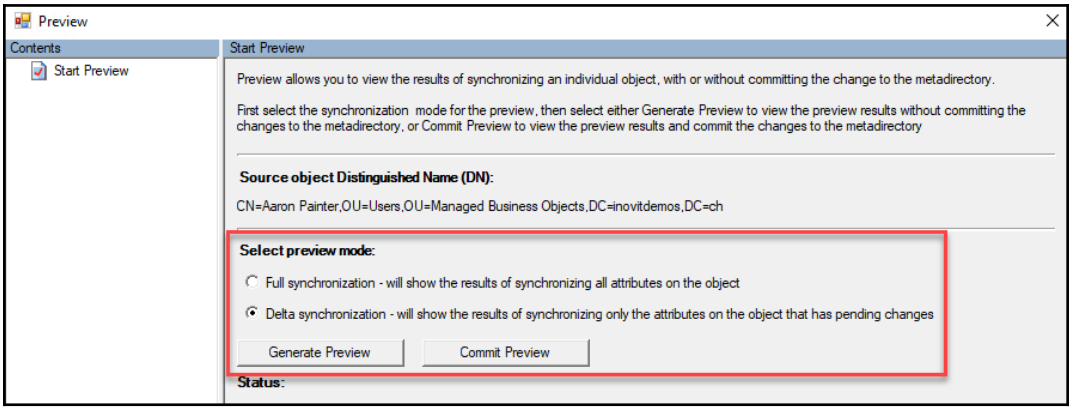
Delta import synchronization statistics

15. Click on **Updates**.
16. You'll notice add on your modified user account:



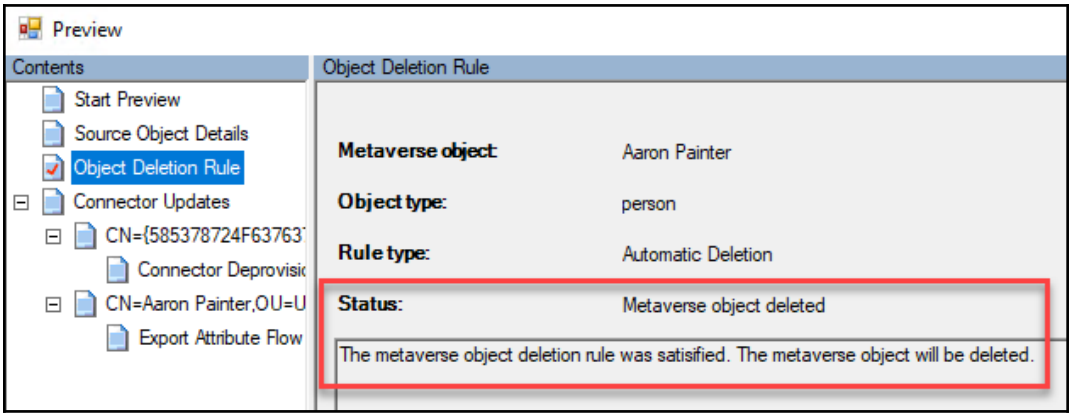
adminDescription attribute flow

17. Click on **Preview**.
18. Choose **Delta Synchronization**.
19. Click **Generate Preview**:



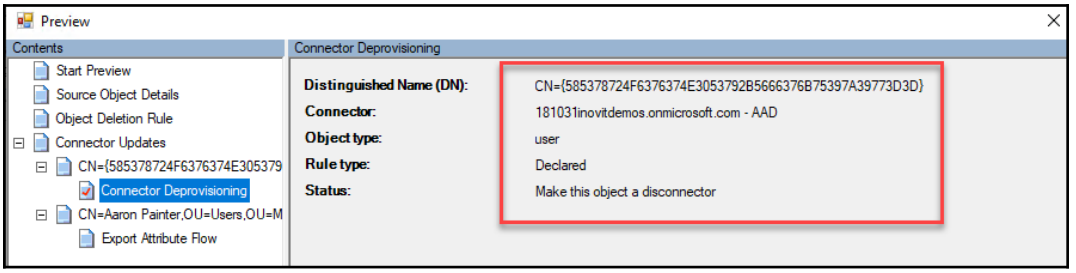
Synchronization preview options

20. You'll see in the result that the **Object Deletion Rule** was satisfied:



Object deletion rule preview

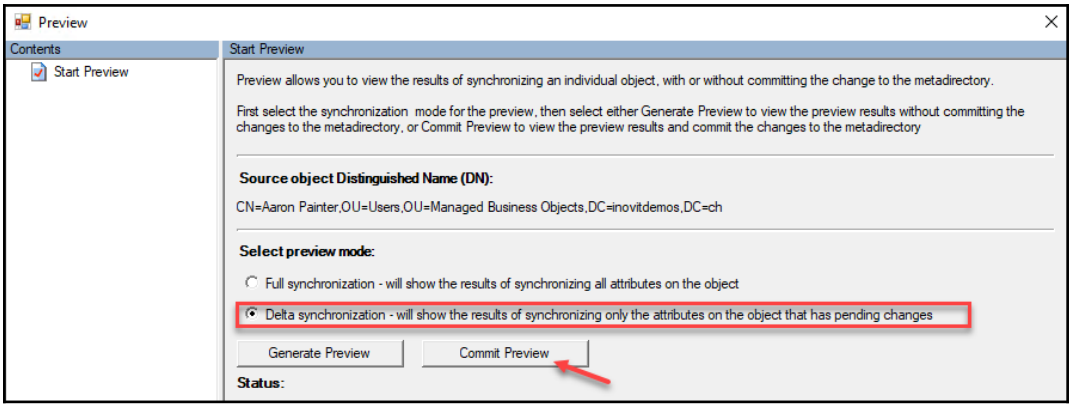
21. Click on **Connector Deprovisioning**:



Connector Deprovisioning preview

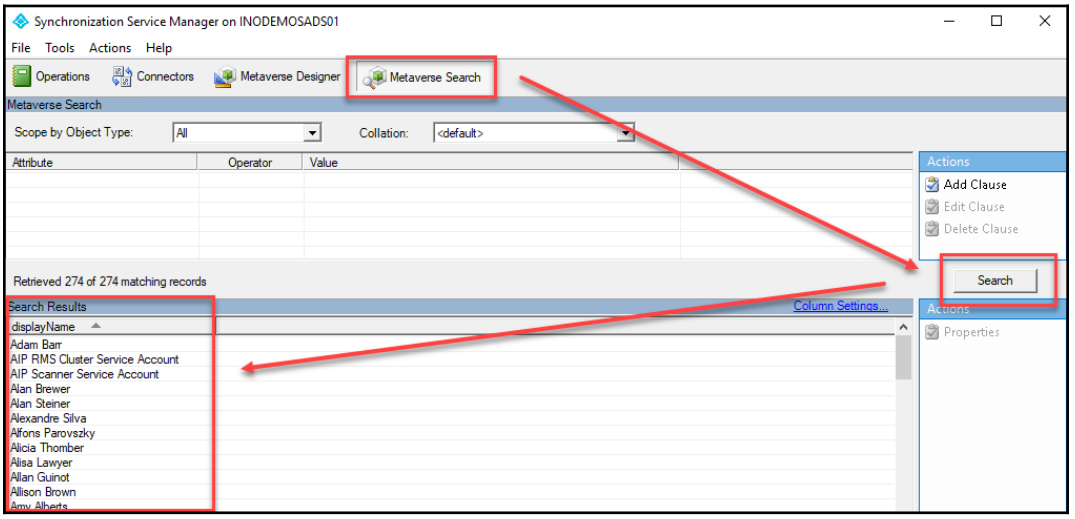
22. The object will be disconnected.
23. Click **Close**.
24. Click **Preview**.
25. Chose **Delta synchronization.**

26. Click **Commit Preview**:



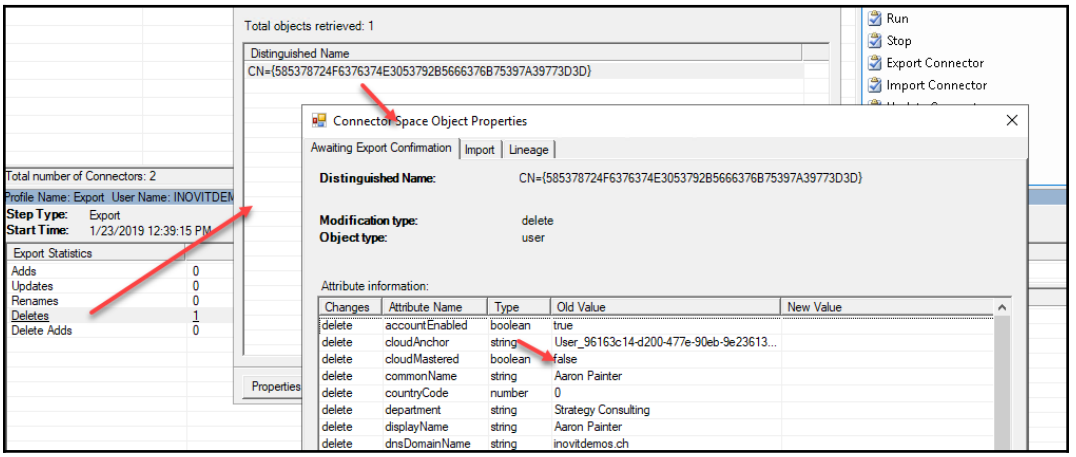
Preview commitment option

27. Open a **Metaverse Search**:



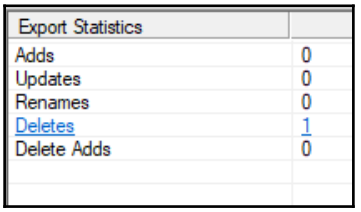
Metaverse search option

28. Click **Search**.
29. Aaron Painter no longer exists in the Metaverse.
30. Run an **Export** on the Azure AD connector:



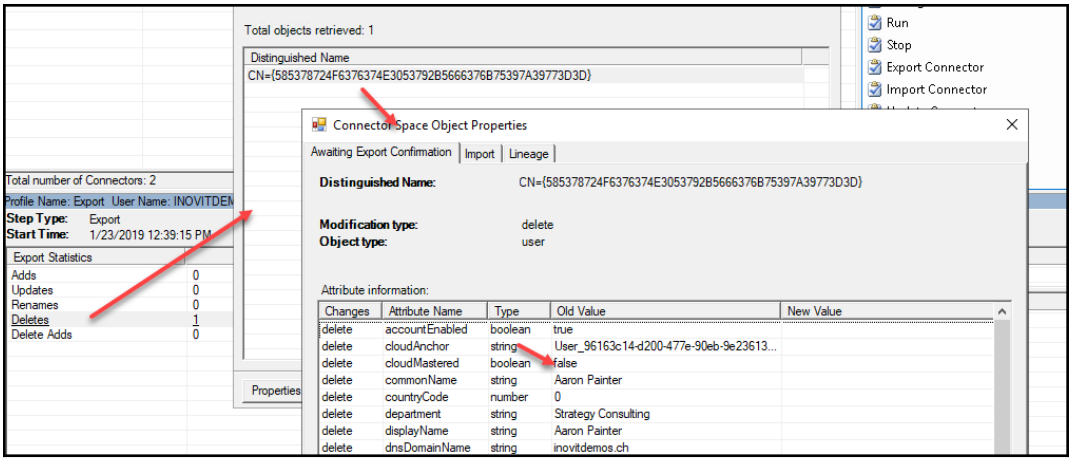
Export run profile execution on Azure AD

31. You'll get a delete in the sync statistics.
32. Click **Deletes**:



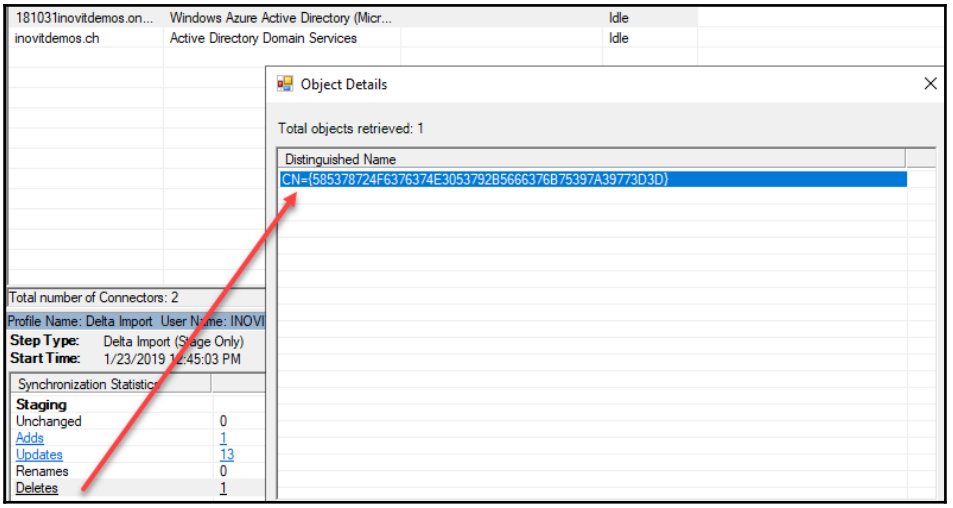
Export Statistics on Azure AD

33. Click on the object GUID:



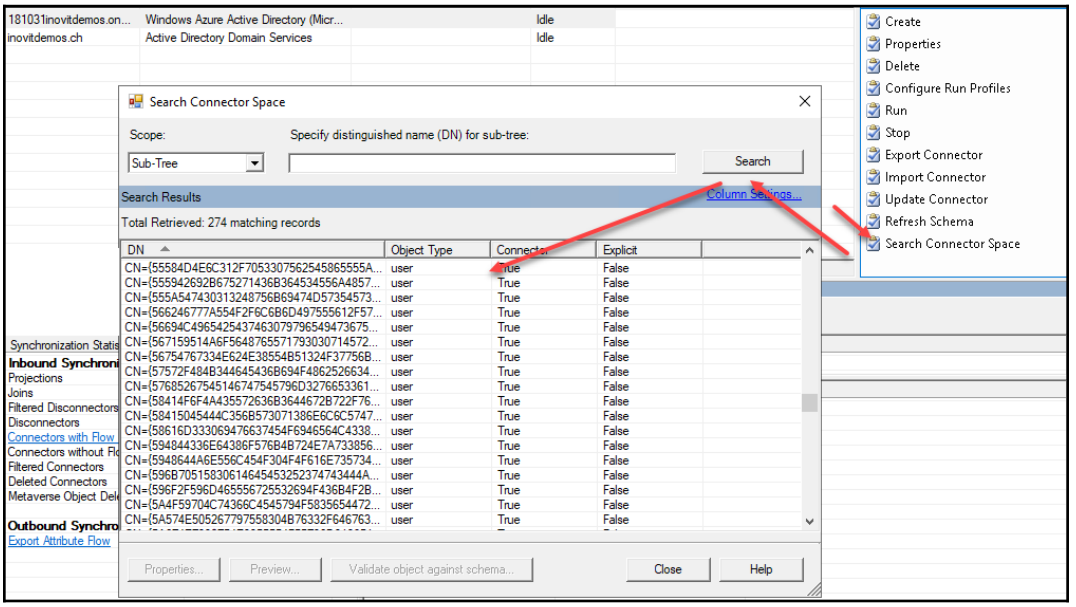
Connector space validation

34. Your user will be deleted from the Azure AD.
35. You'll see Awaiting Export Confirmation.
36. Run delta import on the Azure AD connector.
37. The object will be deleted in the connector space of the Azure AD connector:



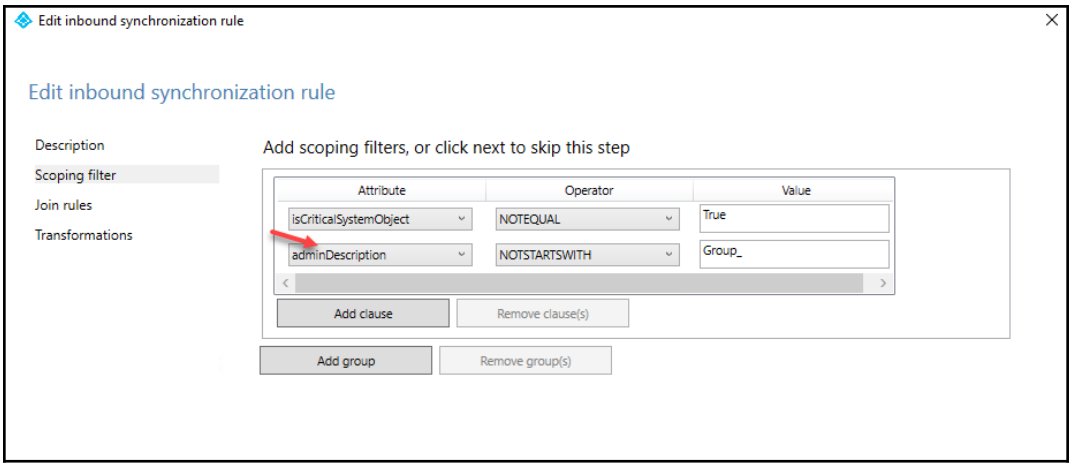
Deletion of the object in the connector space

38. Run delta sync on the Azure AD connector to finish the change.
39. Open the connector space on the Azure AD connector. You won't find the preceding `CN=`:



Connector space search and validation options

With this option, you've seen that you can filter objects in a filter OU by filling the **adminDescription** attribute with **User_NoSync**. The same procedure can be used to filter groups with `Group_NoSync`, for example. Here, you can find the specific `In from AD—Group Common` inbound rule, which is shown in the following screenshot:



adminDescription usage for groups

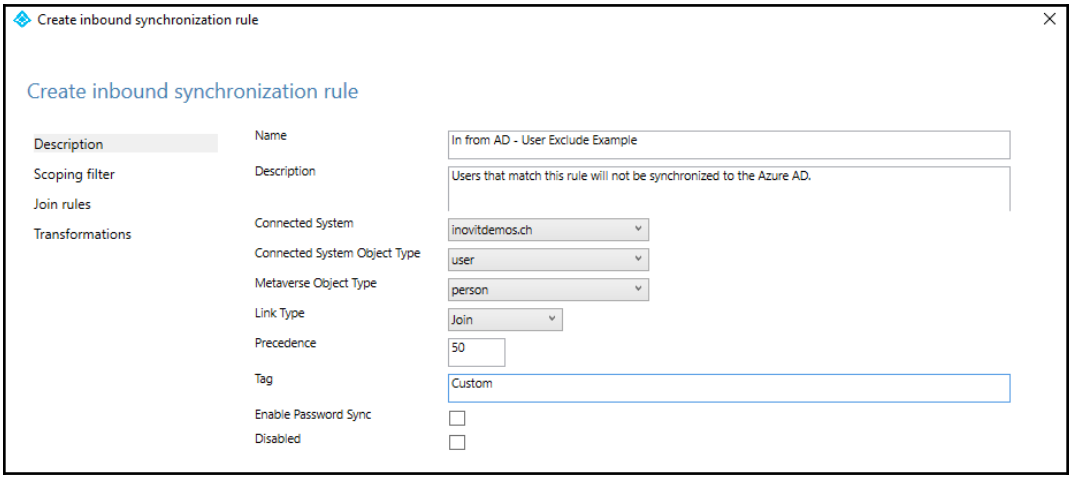
In the next example, we'll build a custom rule.

Building a custom rule for filtering

In this example, we'll build our own rule to filter by a specific department. You can use the following procedure in the **Synchronization Rules Editor**:

1. Open the **Synchronization Rules Editor**.
2. Click on **Add New Rule** and select **Inbound**.

3. Configure the **Description** section:

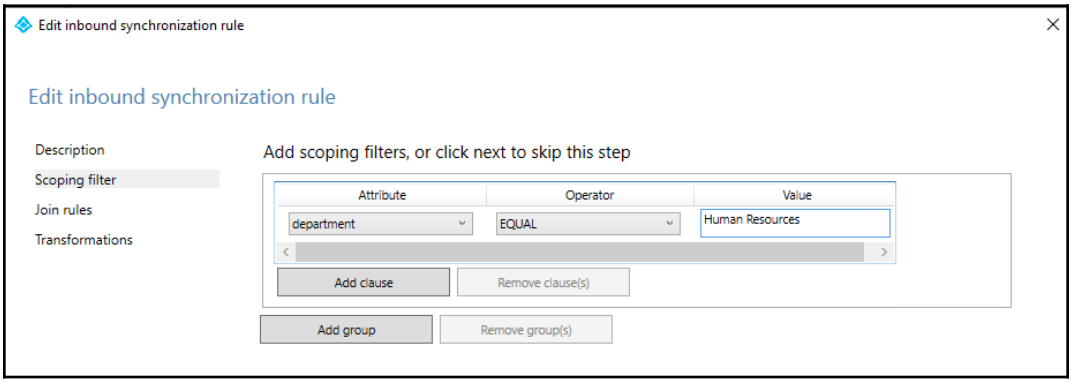


Inbound rule creation process

4. Build a **Scoping filter**:

```
department EQUAL Human Resources
```

5. The configuration should look like this:



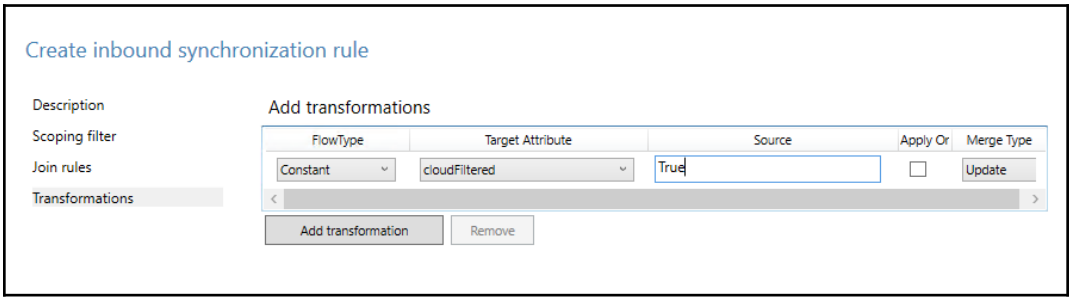
Scope filtering by department attribute

6. Leave the **Join rules**.

7. Go to **Transformations** and create a transformation:

```
Constant cloudFiltered True
```

8. This is shown in the following screenshot:



Cloud filtering option

9. Save the rule and step through the following run profile steps in the synchronization manager:
    - **Delta Import**: AD Connector
    - **Delta Sync**: AD Connector
    - **Export**: AAD Connector
    - **Delta Import**: AAD Connector
    - **Delta Sync**: AAD Connector
10. Verify that the human resources users are deleted in your Azure AD.

With the following example, you can block synchronizing the `thumbnailPhoto` with Azure AD. Furthermore, the configuration will also remove other uploaded `thumbnailPhotos`.

You can use the following PowerShell to configure the scenario:

```
$ThumbnailRules = (Get-ADSyncRule | Where-Object
{($_.AttributeFlowMappings.Source -eq "thumbnailPhoto") -and
($_.Direction -eq "Inbound")})

foreach($syncrule in $ThumbnailRules) {

  Write-Host -ForegroundColor Green "Processing Rule
$($syncrule.Name)"
 $mapping = $syncrule.AttributeFlowMappings | Where-Object {$_.Source
-eq "thumbnailPhoto"}
```

```
  Remove-ADSyncAttributeFlowMapping -SynchronizationRule $syncrule -
AttributeFlowMappings $mapping -OutVariable rule

  Add-ADSyncAttributeFlowMapping -SynchronizationRule $syncrule -
Source @('thumbnailPhoto') -Destination 'thumbnailPhoto' -FlowType
'Expression' -Expression 'AuthoritativeNull' -ValueMergeType 'Update'
-OutVariable rule

  Add-ADSyncRule -SynchronizationRule $rule[0]
}
```

Connecting Azure AD Connect to the second forest

Now that we've seen how the synchronization rules work, we can integrate an additional Active Directory forest to our configuration.

Perform the following steps and use the mentioned scripts to configure your environment:

1. Create the Demo Org Structure:

```
New-ADOrganizationalUnit -Name "Managed Business Objects" -
Path "DC=AZUREID,DC=CH"
New-ADOrganizationalUnit -Name "Users" -Path "OU=Managed
Business Objects,DC=AZUREID,DC=CH"
New-ADOrganizationalUnit -Name "Groups" -Path "OU=Managed
Business Objects,DC=AZUREID,DC=CH"
New-ADOrganizationalUnit -Name "Devices" -Path "OU=Managed
Business Objects,DC=AZUREID,DC=CH"
New-ADOrganizationalUnit -Name "Managed Service Objects" -Path
"DC=AZUREID,DC=CH"
New-ADOrganizationalUnit -Name "AAD" -Path "OU=Managed Service
Objects,DC=AZUREID,DC=CH"
New-ADOrganizationalUnit -Name "Users" -Path
"OU=AAD,OU=Managed Service Objects,DC=AZUREID,DC=CH"
```

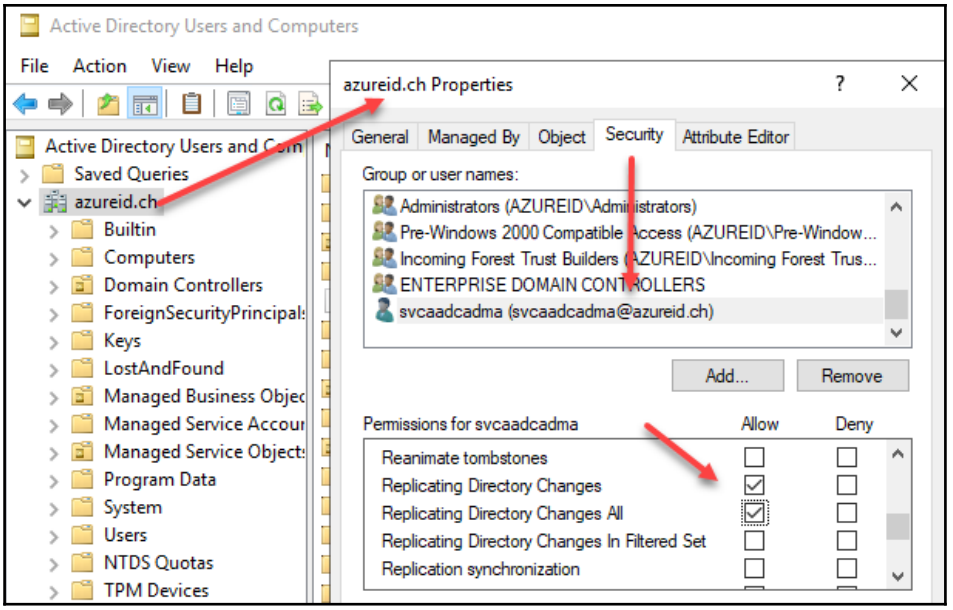
2. Create the Azure AD Connect AD Management Agent service account:

```
New-ADUser -Name "svcaadcadma" -SamAccountName svcaadcadma -
UserPrincipalName svcaadcadma@azureid.ch -path
"OU=Users,OU=AAD,OU=Managed Service Objects,DC=azureid,DC=ch"
-AccountPassword (ConvertTo-SecureString "YOURPASSWORD" -
AsPlainText -Force) -Enabled $True
```

3. Create the **group managed service account** (**gMSA**) to run the AAD Connect:

```
Add-KdsRootKey -EffectiveTime (Get-Date).AddHours(-10)
New-ADServiceAccount -Name svcaadconnect -DNSHostname
INOAZUREIDADS01 -PrincipalsAllowedToRetrieveManagedPassword
INOAZUREIDADS01$
```

4. Set the following rights for the AD Management Agent account on the domain level.
5. Configure the permissions to configure AAD Connect `svcaadcadma`:
    - **Replicate Directory Changes**
    - **Replicate Directory Changes All**:



Active Directory MA service account permissions

6. Enable the AD `Recycle Bin Feature`:

```
Enable-ADOptionalFeature -Identity 'CN=Recycle Bin
Feature,CN=Optional Features,CN=Directory Service,CN=Windows
NT,CN=Services,CN=Configuration,DC=azureid,DC=ch' -Scope
ForestOrConfigurationSet -Target 'azureid.ch'
```

Create your test users with the following in detailed viewed script:

1. Import the Active Directory module:

```
Import-module activedirectory
```

2. The script part auto-populates the currently used domain:

```
$dnsDomain =gc env:USERDNSDOMAIN
 $split = $dnsDomain.split(".")
 if ($split[2] -ne $null) {
 $domain = "DC=$($split[0]),DC=$($split[1]),DC=$($split[2])"
 } else {
 $domain = "DC=$($split[0]),DC=$($split[1])"
 }
```

3. Declare any variables:

```
$dirpath = $pwd.path
 $orgUnit = "OU=Managed Business Objects"
 $dummyPassword = ConvertTo-SecureString -AsPlainText
"OnBo@rdInoDemos19!" -Force
 $counter = 0
```

4. Import the CSV file:

```
$ImportFile = Import-csv "$dirpath\ADUsers.csv"
 $TotalImports = $importFile.Count
```

5. Create users:

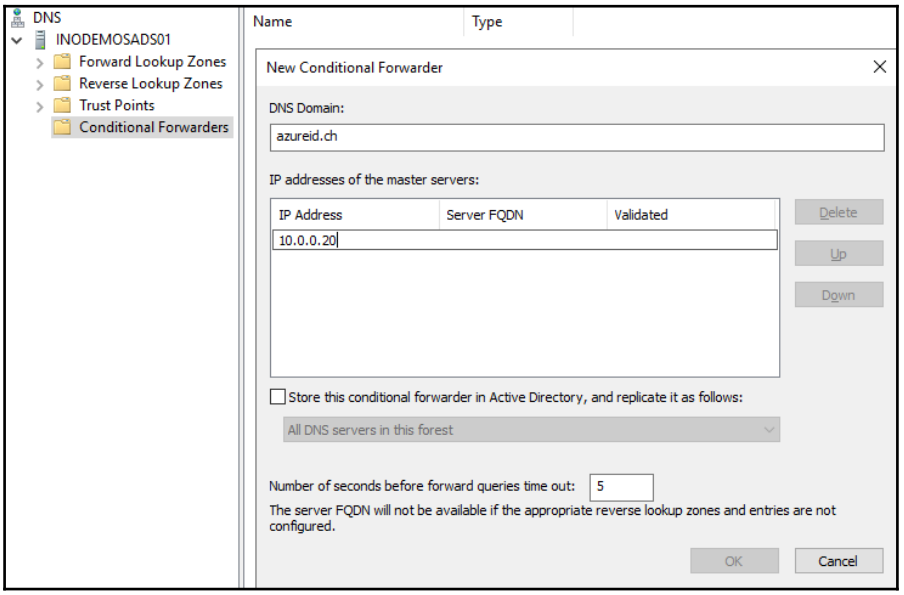
```
$ImportFile | foreach {
 $counter++
 $progress = [int]($counter / $totalImports * 100)
 $upn = "$($_.GivenName).$($_.Sn)@azureid.ch"
 Write-Progress -Activity "Provisioning User Accounts" -status
"Provisioning account $counter of $TotalImports" -perc
$progress
 if ($_.Manager -eq "") {
 New-ADUser -SamAccountName $_.SamAccountName -Name $_.Name -
Surname $_.Sn -GivenName $_.GivenName -Path "$orgUnit,$domain"
-AccountPassword $dummyPassword -Enabled $true -title $_.title
-officePhone $_.officePhone -department $_.department -
UserPrincipalName $upn -PasswordNeverExpires $false
 } else {
 New-ADUser -SamAccountName $_.SamAccountName -Name $_.Name -
Surname $_.Sn -GivenName $_.GivenName -Path "$orgUnit,$domain"
-AccountPassword $dummyPassword -Enabled $true -title $_.title
```

```
-officePhone $_.officePhone -department $_.department -manager
"$($_.Manager),$orgUnit,$domain" -UserPrincipalName $upn -
PasswordNeverExpires $false
 }
 If (gci "$dirpath\userimages\$($_.name).jpg") {
 $photo =
[System.IO.File]::ReadAllBytes("$dirpath\userImages\$($_.name)
.jpg")
 Set-ADUSER $_.samAccountName -Replace
@{thumbnailPhoto=$photo}
 }
 }
```

Create the email addresses from `userPrincipalName` with the following script:

```
 Import-Module ActiveDirectory
 Get-ADUser -LDAPFilter '(userPrincipalName=*)' `
-Properties userPrincipalName,mail | Select-Object * | `
ForEach-Object { Set-ADObject -Identity `
$_.DistinguishedName -Replace `
@{mail=$($_.userPrincipalName)} }
```

Create a DNS forwarder to your other domain, because Active Directory trust has been established:



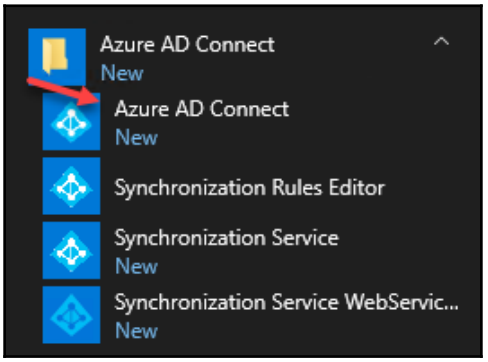
DNS configuration for additional forests

Configure the `ms-ds-consistencyGuid` permissions for the `svcaadcadma` service account:

```
$accountName = "AZUREID\svcaadcadma"
$ForestDN = "DC=AZUREID,DC=CH"
$cmd = "dsacls '$ForestDN' /I:S /G '`"$accountName`":WP;ms-ds-
consistencyGuid;user'"
Invoke-Expression $cmd
```

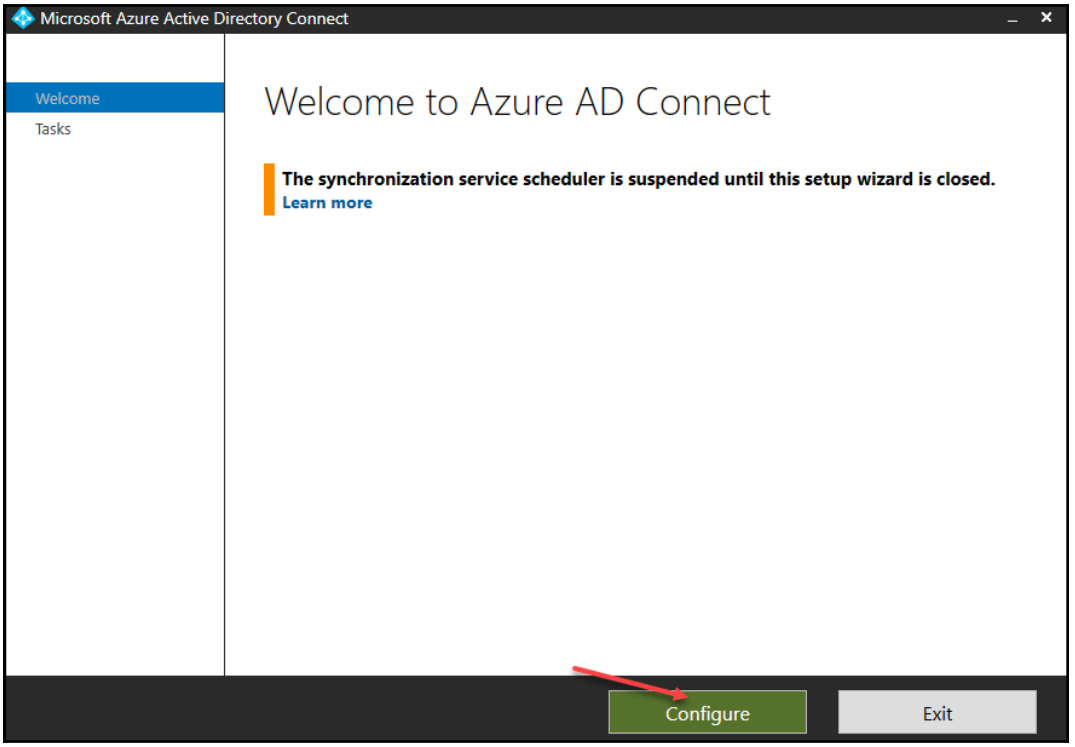
Now that we've prepared our second Active Directory Forest, we can include it in our Azure AD Connect:

1. Open the Azure AD Configuration wizard on the **YD1ADS01** server:



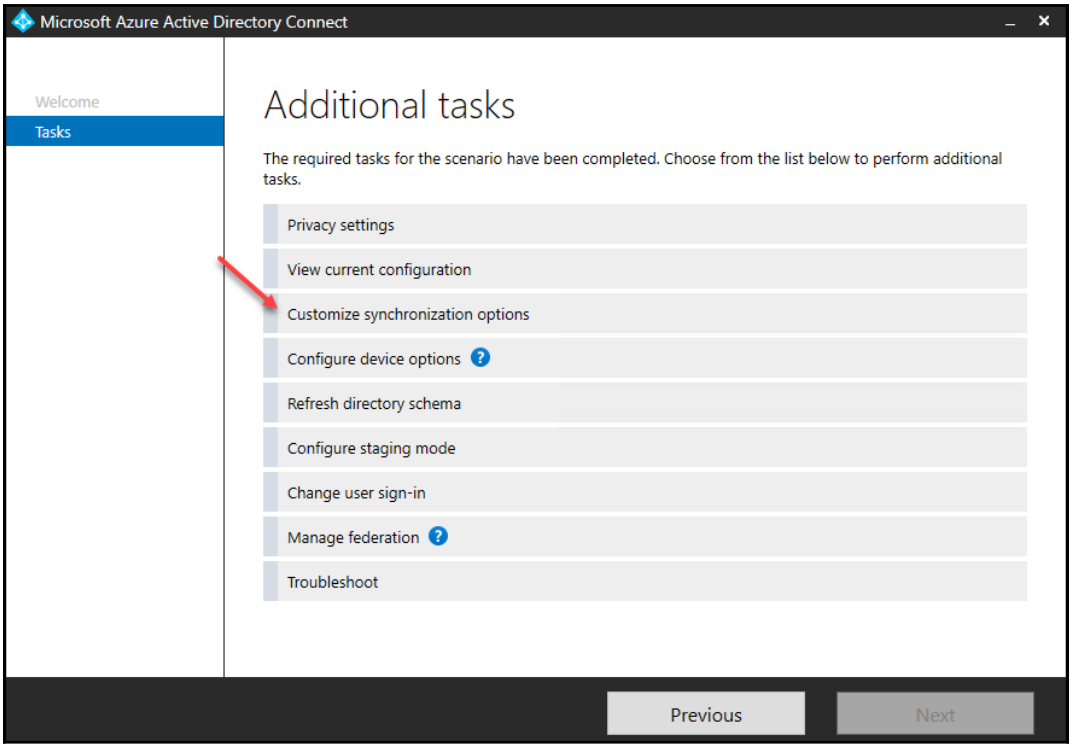
Start Menu–Azure AD Connect

2. Click **Configure**:



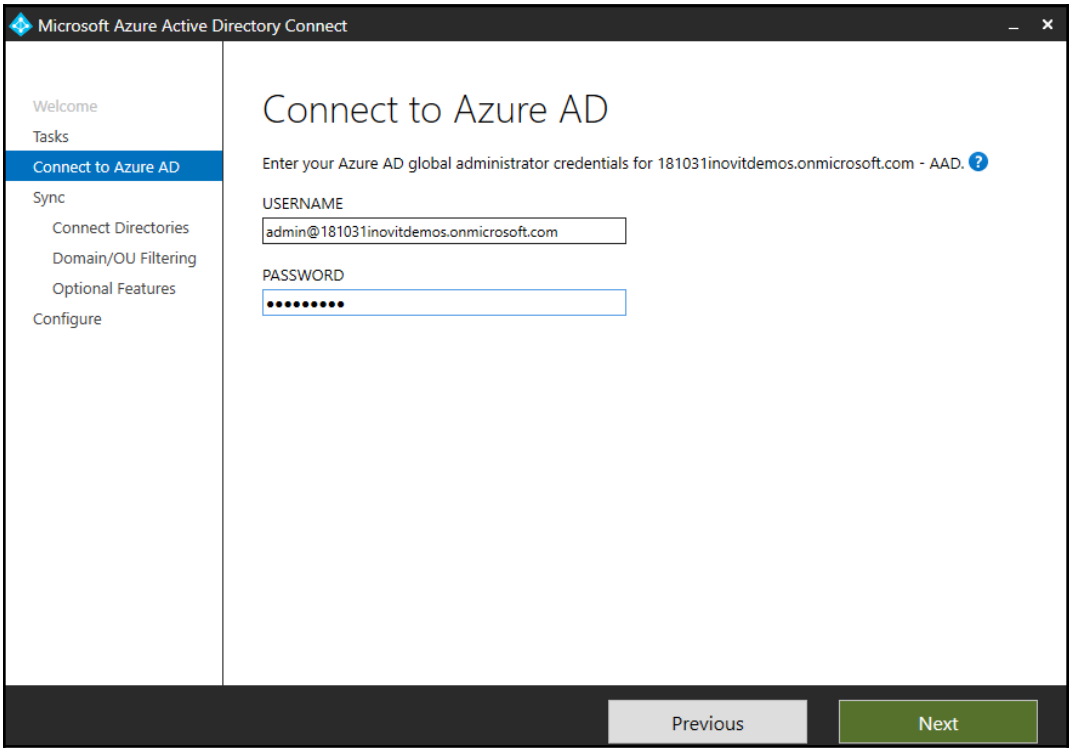
Azure AD Connect configuration

3. Click **Customize synchronization options**:



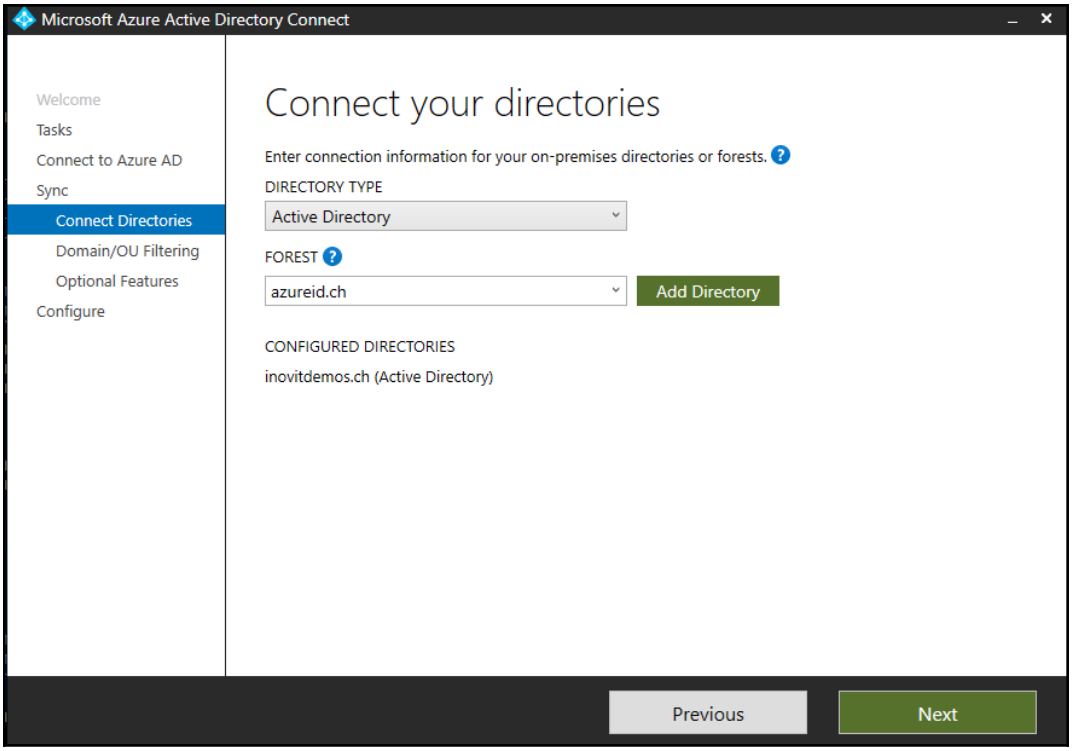
Synchronization options customization

4. **Connect to Azure AD** with **global administrator** rights:



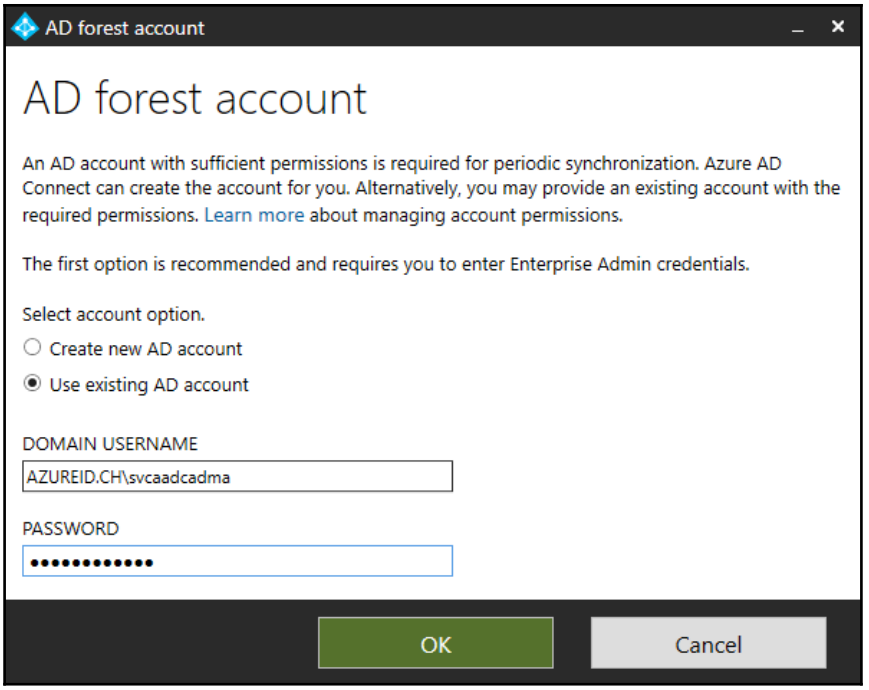
Providing global administrator credentials for Azure AD connection

5. Connect the second forest:



Connecting the additional AD forest

6. Use the service account in the other forest to connect:

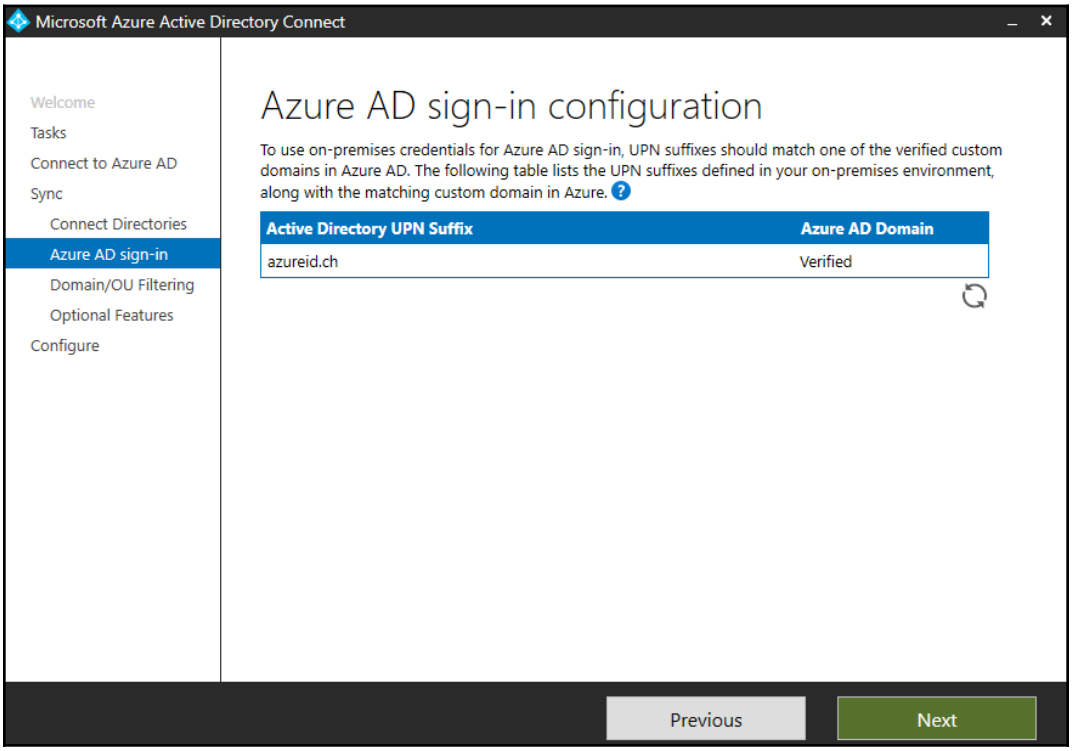


Using the AD Management Agent service account of the additional forest

7. You should get a second AD that's valid in your list:

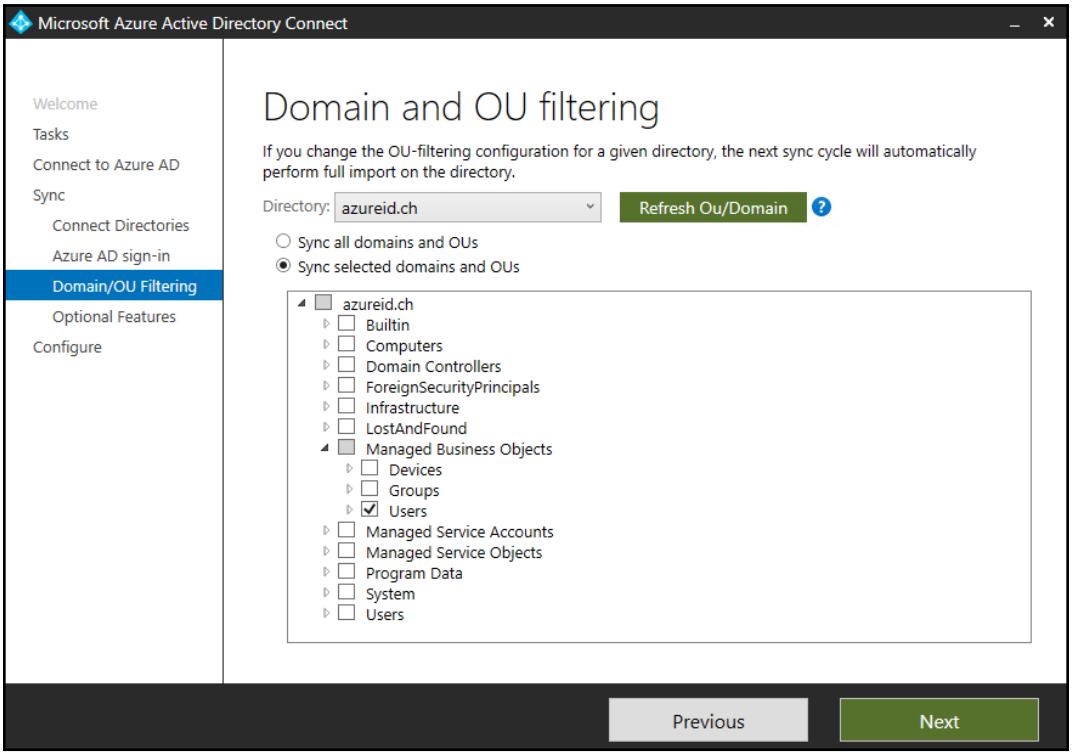
Valid connection overview

8. As we already verified the domain suffixes, our domain suffix is okay to sync:



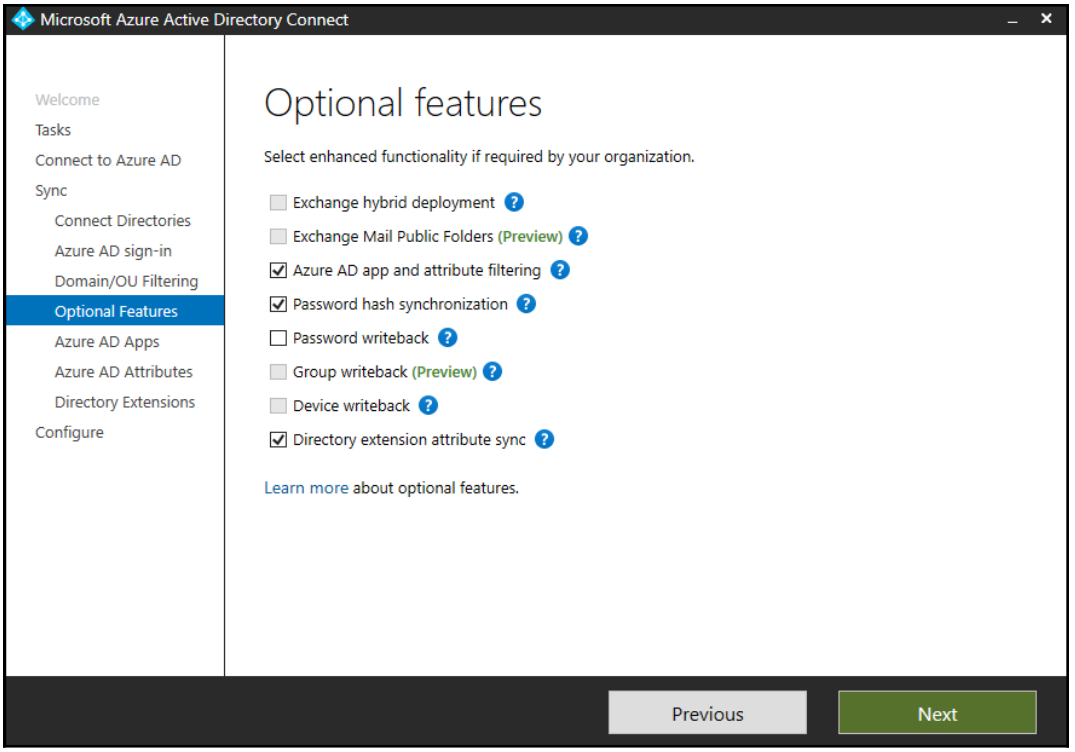
Sign-in configuration option

9. Filter the **OU** you want to synchronize:



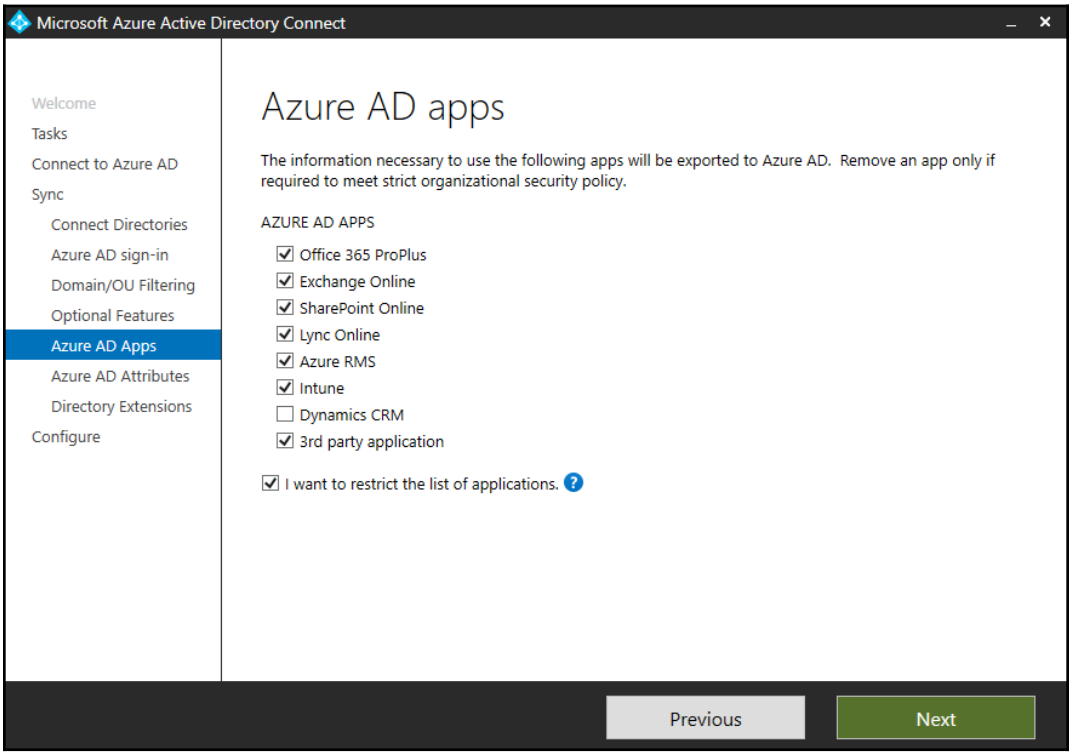
Filtering the related objects

10. Enable the following features to see the extended synchronization options:



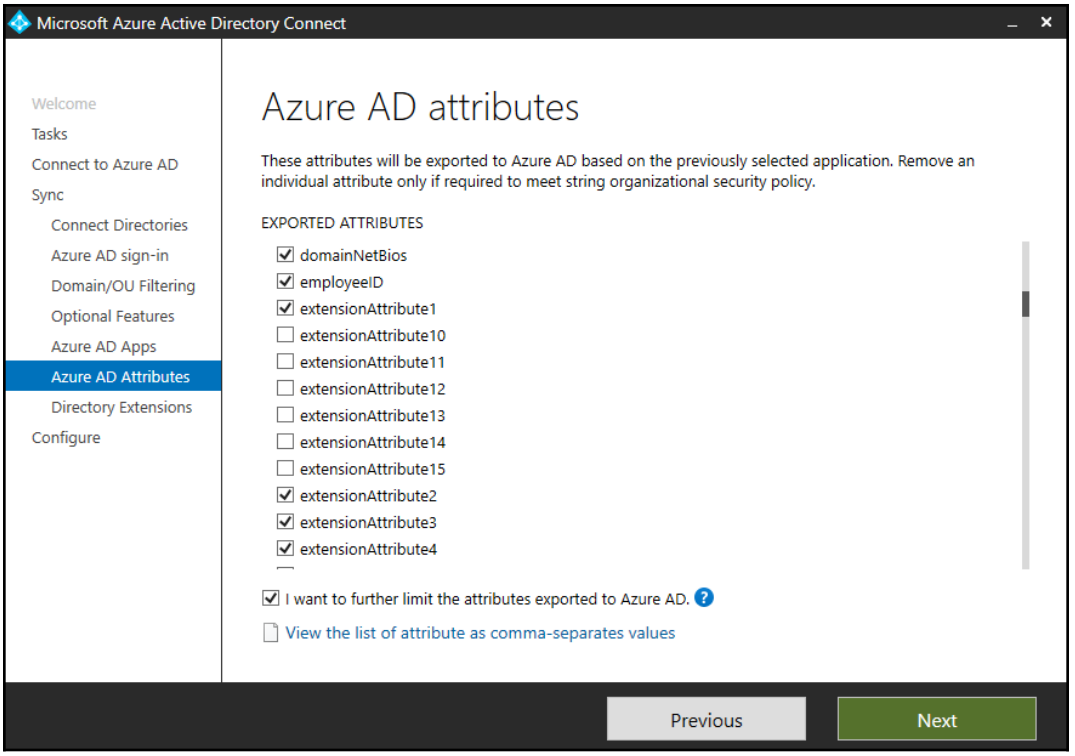
Configuration of optional features—app and attribute filtering, password hash sync, and extension attributes

11. With this configuration, you can limit the attributes that get synced to Azure AD for specific workloads:



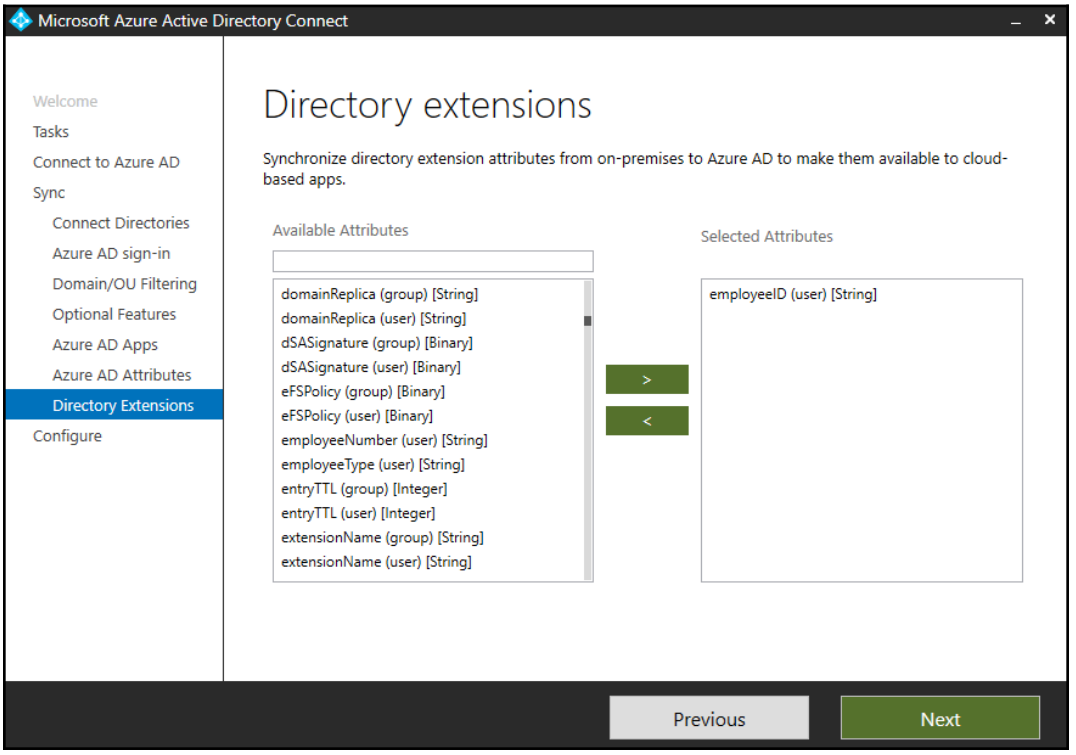
App filtering options

12. Choose which attributes get exported:



extensionAttribute limitations

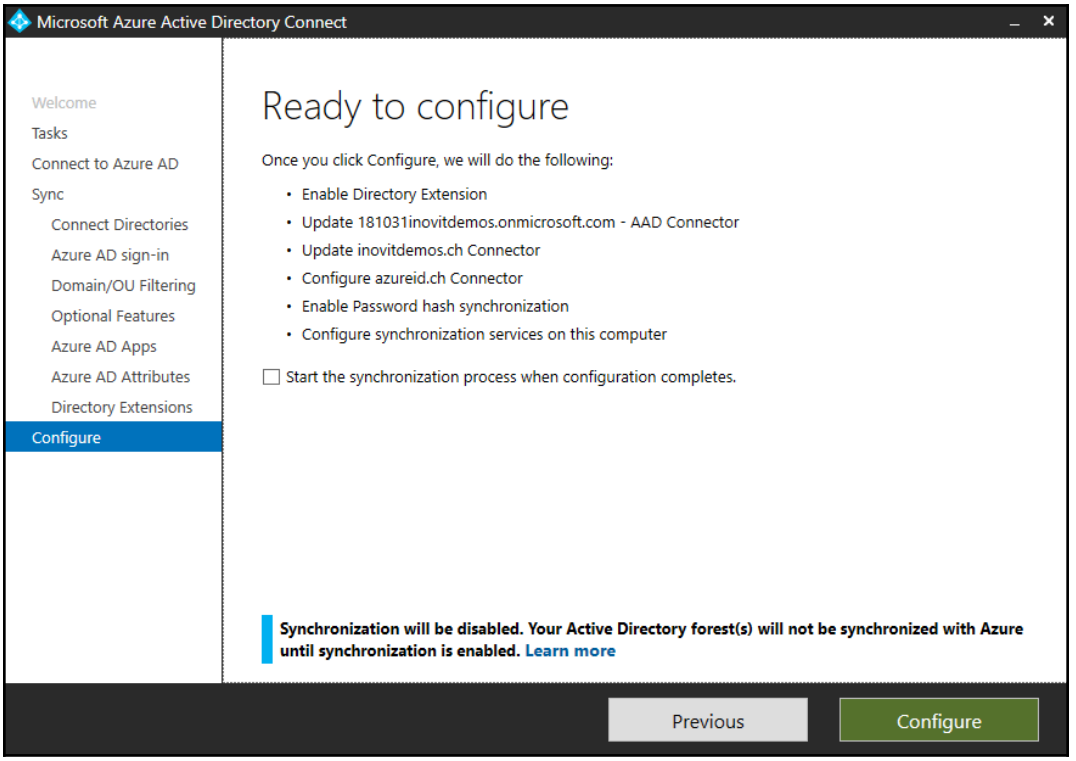
13. Choose some additional attributes, but keep in mind that not all services will get these attributes:



Directory extension implementation

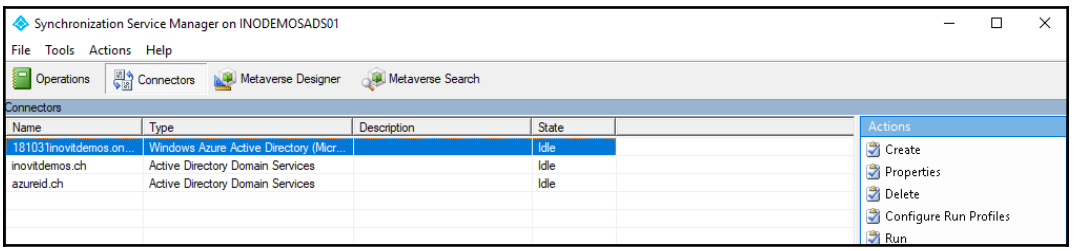
Never start the synchronization directly—Initial load is always a manual stepping process that's used to avoid errors.

14. We're ready to start the configuration, so click **Configure**:



Final configuration step

15. Close the configuration assistant.
16. Let's start stepping through the initial load of the new Active Directory forest, so open the **Synchronization Service Manager**:



Connected directories overview

17. Start the initial load with the following steps—and control every step:
    1. Run a full import on all AD Forests
    2. Run a full import on AAD
    3. Run a full sync on all AD Forests
    4. Run a full sync on Azure AD
    5. Run an Export on AAD
    6. Run an Export on all AD Forests

Yeah! We connected our new forest under complete control. Congrats!

The authentication part for this will be described in `Chapter 7`, *Deploying Solutions on Azure AD and ADFS*, `Chapter 8`, *Using the Azure AD App Proxy and the Web Application Proxy*, and `Chapter 10`, *Exploring Azure AD Identity Services*.

Keep an eye out for the ADFS B2B configuration with a claim's provider trust in `Chapter 10`, *Exploring Azure AD Identity Services*.

Summary

In this chapter, you learned about declarative provisioning and how synchronization rules work. We explored practical examples to learn how to use standard sync rules and how to create our own. You also learned how to use PowerShell to configure synchronization rules and block/delete attributes to and in Azure AD. With the integration of another untrusted Active Directory, you saw the standard procedure to do this and to customize your synchronization options to define precisely what should happen in your environment.

In the next chapter, we'll show you how to monitor your synchronization solution.

4
Monitoring Your Identity Bridge

Monitoring your identity synchronization processes, your Active Directory health status, and the functionality of your ADFS and Web Application Proxy authentication platform is essential for your organization. Also, gathering performance data is necessary to provide a suitable infrastructure.

In this practical chapter, we'll explore the various monitoring capabilities for the identity bridge that's constructed by Azure AD Connect, the Active Directory itself and, if used, the ADFS and the Web Application Proxy. We'll investigate the Azure AD Monitoring and logs functionalities, the Azure AD Connect Health service, and the Azure Security Center to get a good idea of several use cases to provide an efficient and accurate monitoring to deliver a stable and suitable identity infrastructure for connecting to Azure AD.

This chapter will cover the following topics:

- How Azure AD Connect Health works
- Azure AD Monitoring and logs
- Azure Security Center for monitoring and analytics

How Azure AD Connect Health works

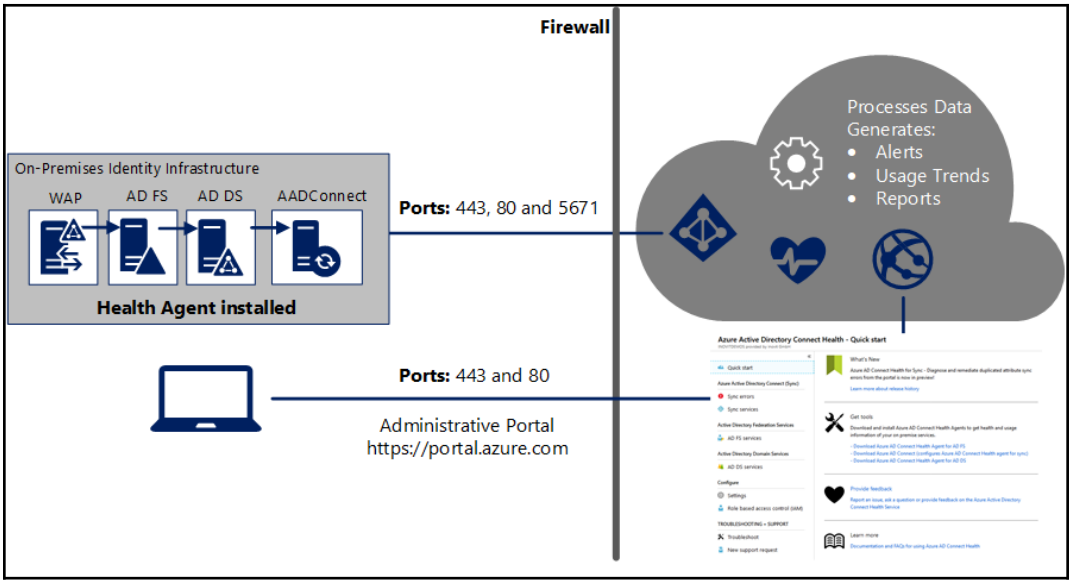
Azure AD Connect Health offers you the ability to monitor and gain insights into the identity infrastructure used to extend on-premises identities to Azure Active Directory and Office 365. You can view alerts, performance information, usage patterns, and configuration settings; it enables you to maintain a reliable connection to Office 365 and much more. This is accomplished by using an agent that's installed on the targeted servers.

The diagram that follows shows how Azure AD Connect Health communicates. It also shows the components that are currently supported:

- **Active Directory Domain Services** (**ADDS**) 2008R2 up to 2016
- Azure AD Connect Version 1.0.9125 or higher
- **Active Directory Federation Services** (**ADFS**) and **Windows Azure Pack** (**WAP**) 2008R2 up to 2016

You can find the actual support and licensing requirements at `https://docs.microsoft.com/en-us/azure/active-directory/hybrid/reference-connect-health-faq`.

The following diagram shows the architecture of the solution:



To deploy all Azure AD Connect Health Agents, you'll need to finish the lab tasks in `Chapter 8`, *Developing Solutions on Azure AD and ADFS*, because in this lab, you'll install ADFS and the Web Application Proxy.

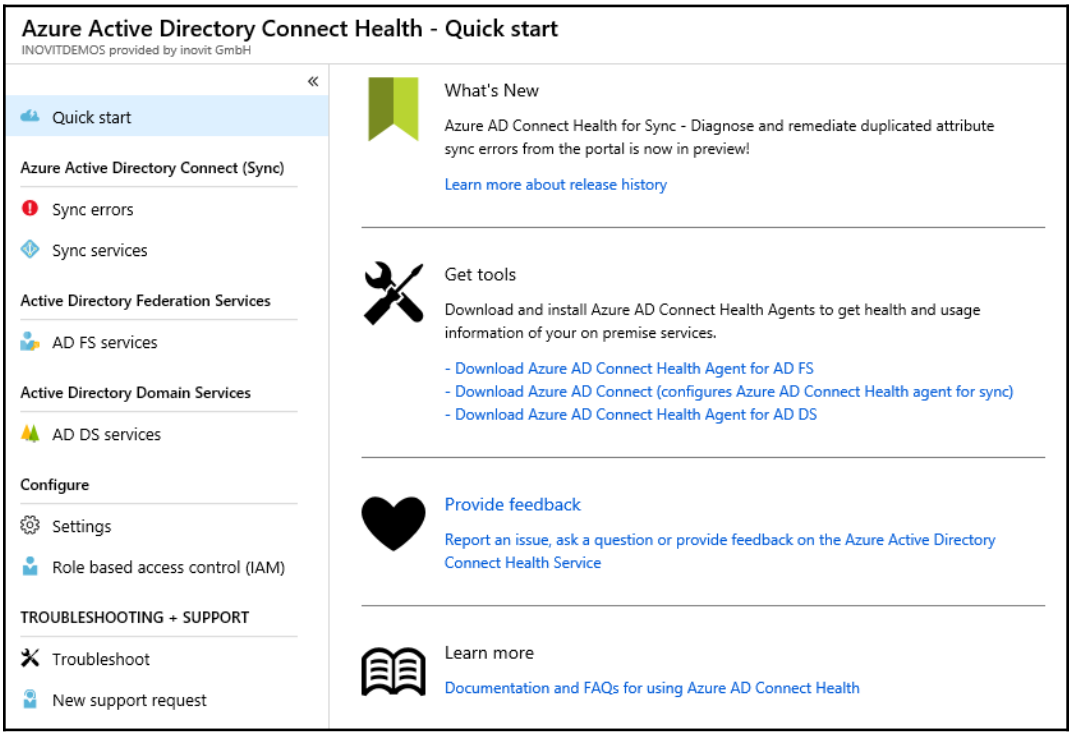
The installation is a very easy task and always follows the same routine. In the following screenshot, under **Get tools**, you'll find all of the actual binaries for the agents.

The health agent for Azure AD Connect will always be installed with the tool itself. Azure AD Connect Health requires Azure AD Premium P1 licenses.

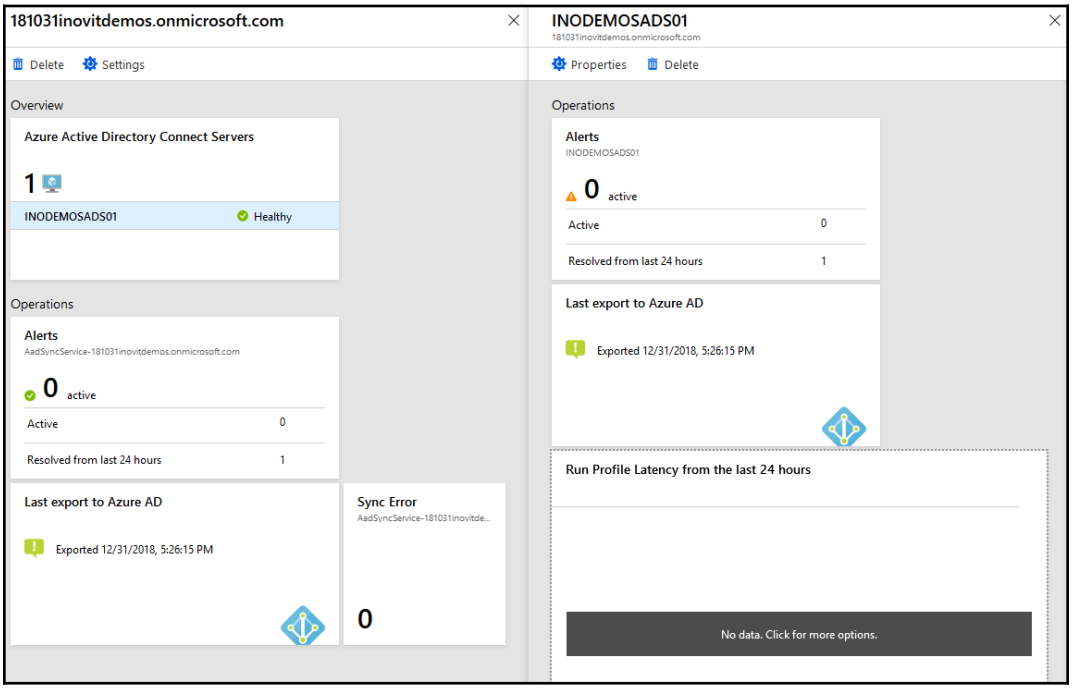
The following steps can be used to install and configure the feature:

1. Download and install the binaries on the specific server.
2. Click **Configure**.
3. Provide global administrator credentials to successfully register the agent with the service.

The following screenshot shows the starting page, where you can download the agents:



With the Azure AD Connect Health Agent, you'll get all of the insights about your synchronization process, as we discussed in `Chapter 2`, *Understanding Identity Synchronization*. You'll find the current configuration information and alerts from the synchronization engine, including synchronization errors. Furthermore, if your Azure AD Connect manages thousands of users, groups, and other objects, you'll get the related information about the **Run Profile latency**. The service helps you to gather the actual performance and values for future planning, as can be seen in the following screenshot:

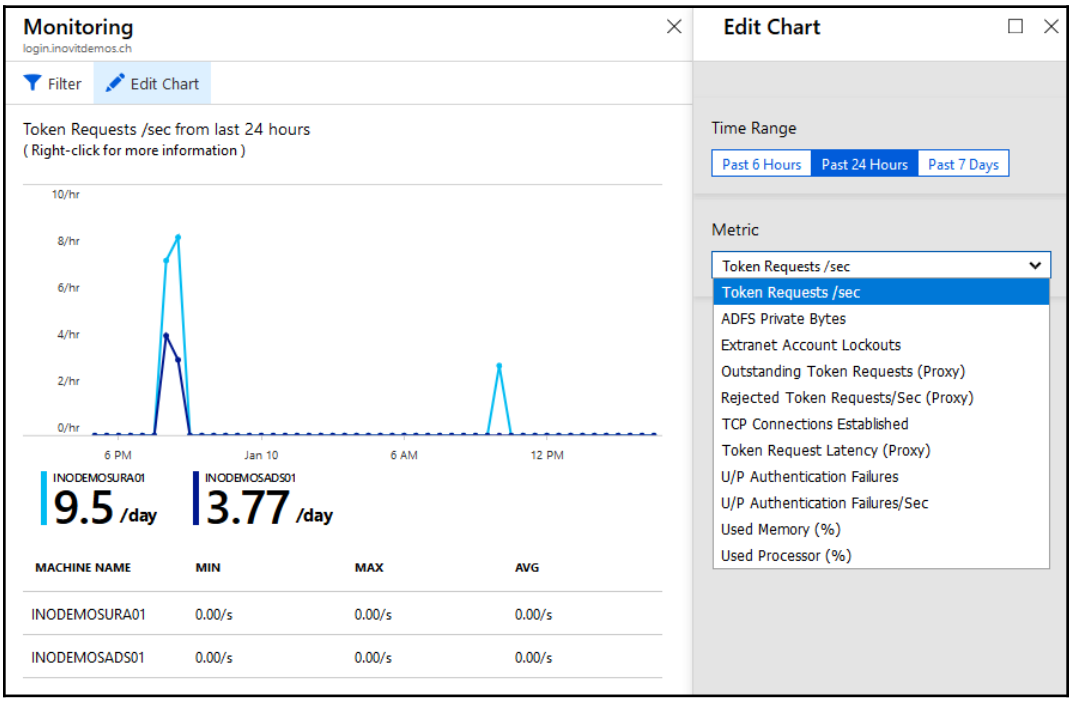


The ADFS Health Agent is used by ADFS and the Web Application Proxy role. The agent needs to be installed on any server with these specific roles. The agent delivers the following fundamental values:

- Security reports for the top 50 users who have bad username/password attempts
- High-failure IP report for ADFS sign-ins
- Monitoring with alerts email notifications

- Usage analytics for ADFS sign-ins with pivots (apps, network location, and other information)
- **Token Requests/sec from the last 24 hours**
- Trends in performance data for capacity planning
- Certificate information

You can see metric values under the chart's configuration in the following screenshot:



You'll find the actual capabilities at `https://docs.microsoft.com/en-us/azure/active-directory/hybrid/how-to-connect-health-adfs`.

The Agent Installation Wizard guides you through a very fast setup experience, and you just need to provide global administrator credentials to register the agent.

If IE enhanced security is enabled, the following websites must be allowed on the server that's going to have the agent installed:

- `https://login.microsoftonline.com`
- `https://secure.aadcdn.microsoftonline-p.com`
- `https://login.windows.net`
- The IE local intranet zone needs to include your ADFS server, such as `https://login.inovitdemos.ch`

The following screenshot shows the installation dialog of the agent:

To use the ADFS Health agent, you need to configure the auditing settings on the server with the following commands:

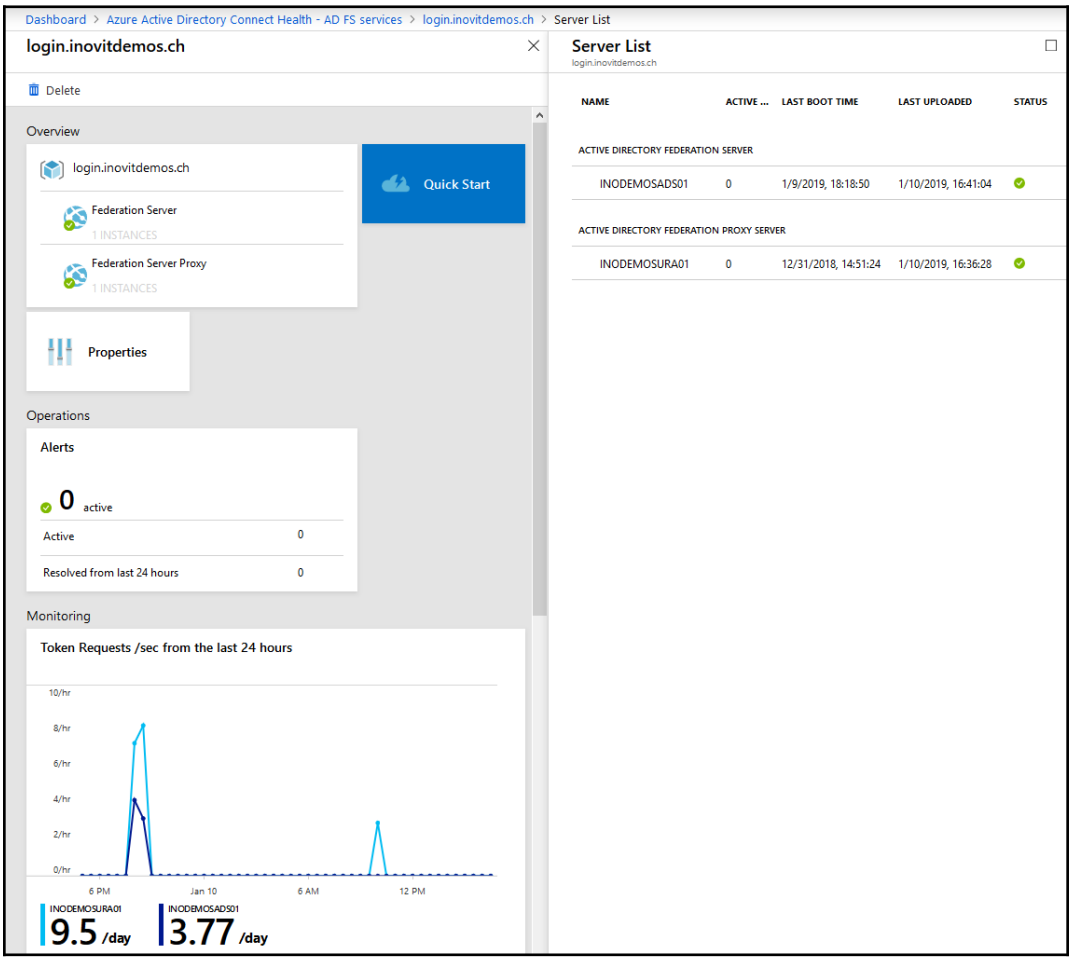
```
auditpol.exe /set /subcategory:"Application Generated" /failure:enable
/success:enable
Set-AdfsProperties –AuditLevel Verbose
```

The ADFS service account needs to able to **Generate security audits**, for example, in the local security policy configuration under **Security Settings**|**Local Policies**|**User Rights Assignment**.

After installing the ADFS Health Agent, you should find the following three new services:

- **Azure AD Connect Health AD FS Diagnostic Service**
- **Azure AD Connect Health AD FS Insight Service**
- **Azure AD Connect Health AD FS Monitoring Service**

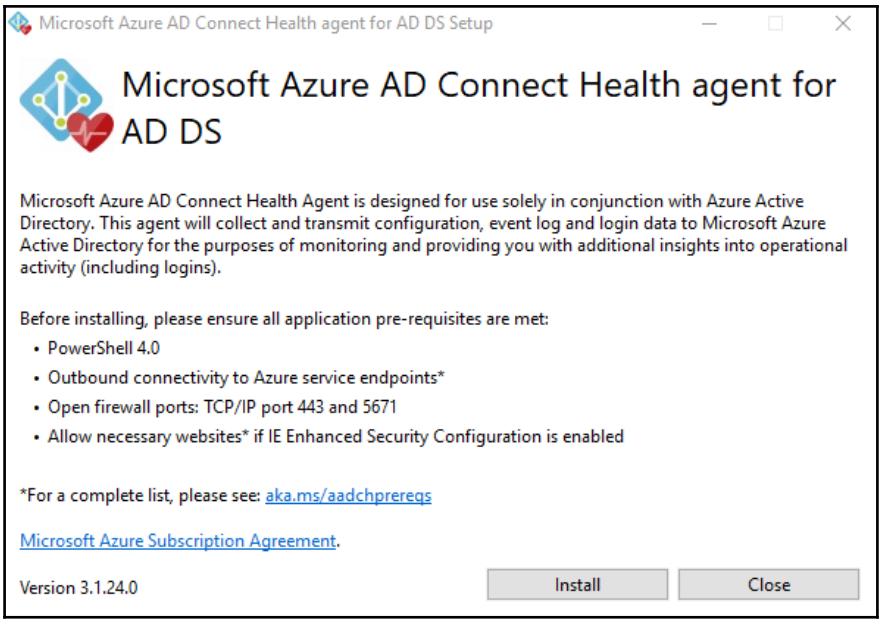
We'll also get a lot of information about the ADFS and Web Application Proxy infrastructure. After completing your configuration in the test environment, we expect a result similar to the following screenshot:



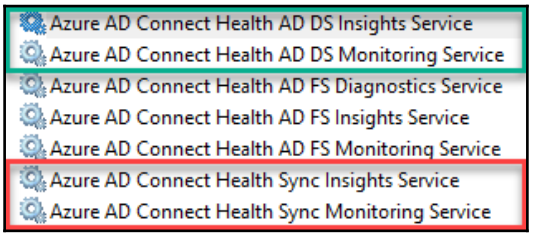
At the time of writing, two new features are the **Bad password attempt** and the **Risky IP** features, which help you to protect against attacks to your authentication infrastructure. In `Chapter 7`, *Deploying Solutions on Azure AD and ADFS*, we'll discuss more security features that will be provided through ADFS on the Windows Server 2019 and the upcoming features in the field of identity protection:

You'll find more information about the feature sets of Azure AD password protection and risky sign-in reports at `https://docs.microsoft.com/en-us/azure/active-directory/authentication/howto-password-ban-bad-on-premises-troubleshoot` and `https://docs.microsoft.com/en-us/azure/active-directory/reports-monitoring/concept-risky-sign-ins`.

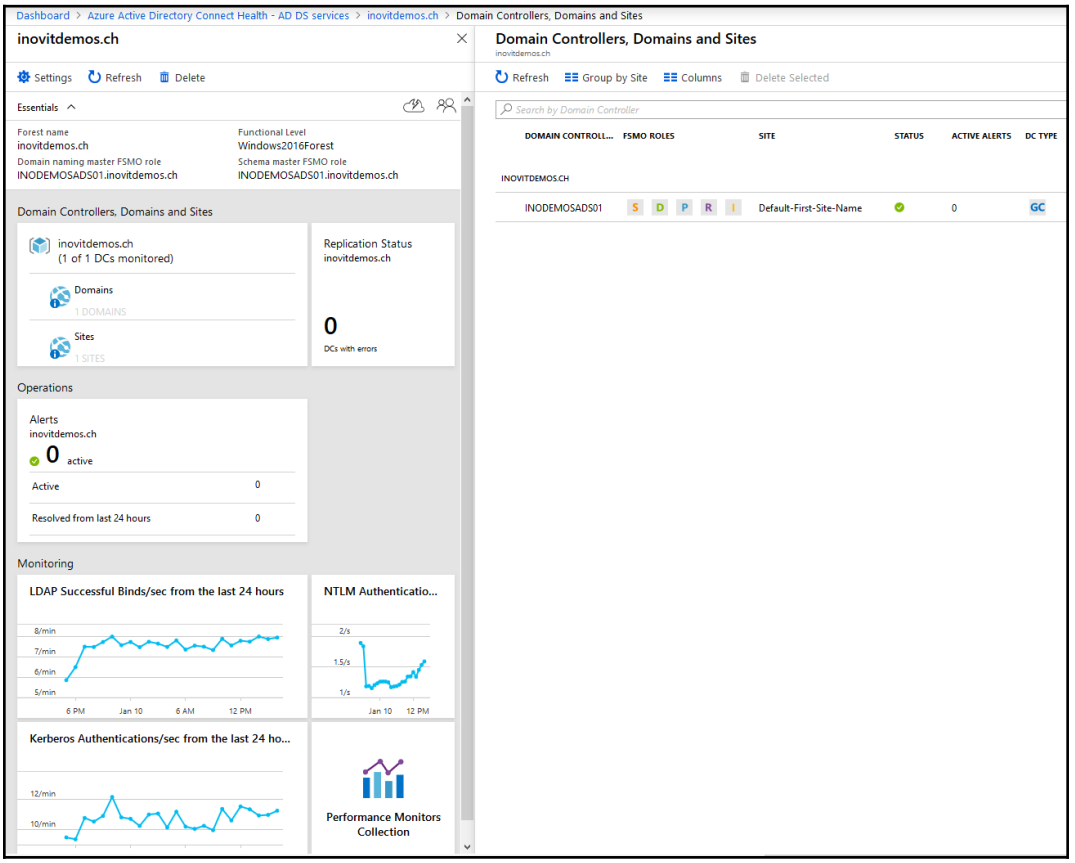
The **Azure AD Connect Health agent** for the Active Directory helps you to identify problems with your directory services infrastructure. You can gather information such as the count of domain controllers, the domains and sites, and the current holders of the **Flexible Single Master Operation** (**FSMO**) roles. A very helpful feature provides the monitoring sections, which allow you to get insights about the LDAP binds and the Kerberos/NTLM authentication. Furthermore, replication information is also provided through this agent. The agent installation follows the same procedures as for ADFS, as you can see in the following screenshot:



After the installation, you should find two more services, as shown at the top of the following screenshot. The last two services listed in the screenshot show the **Azure AD Connect Health Synchronization** services:



The following screenshot shows you insights about the Active Directory health state, including authentication performance data:



To conclude, let's look at some additional practical tips we always need in our projects. The following steps can be useful to configure your environment for several needs:

- **Proxy servers**:
    - You can import the IE settings with the following cmdlet:

```
Set-AzureAdConnectHealthProxySettings -
ImportFromInternetSettings
```

- Or you can import the proxy settings from your WinHTTP configuration:

```
Set-AzureAdConnectHealthProxySettings -
ImportFromWinHttp
```

- Or you can set the proxy completely manually:

```
Set-AzureAdConnectHealthProxySettings -
HttpsProxyAddress address:port
```

- **Testing the health agent connectivity**:
  - Testing can be done with the following cmdlet:

```
Test-AzureADConnectHealthConnectivity -Role ADFS
```

The output for a working configuration looks as follows:

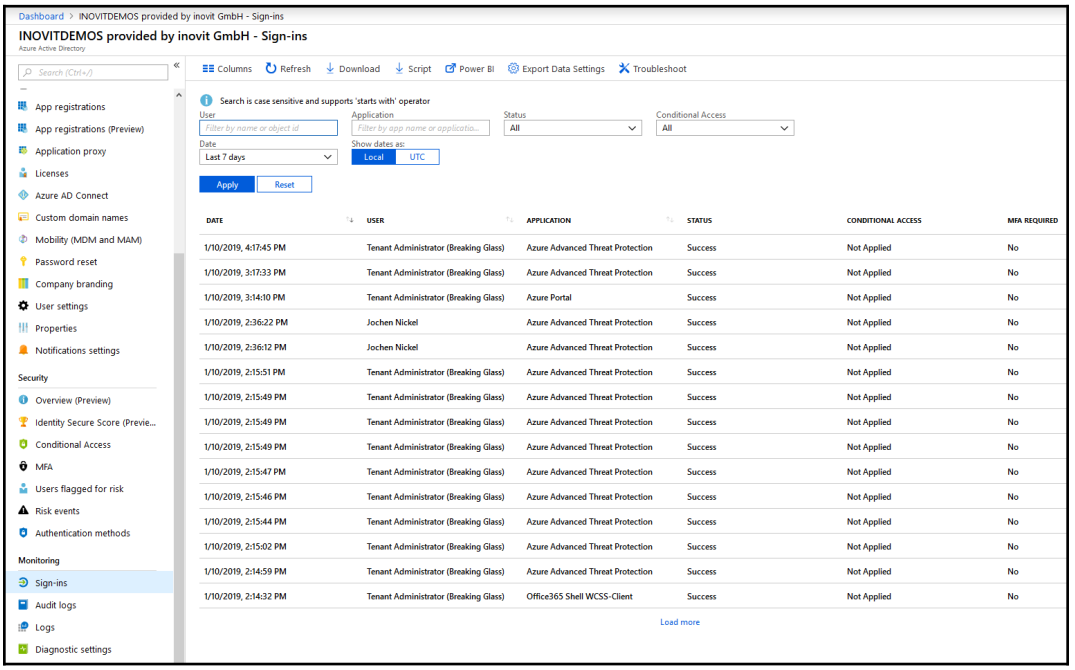
```
PS C:\Users\cloudadmin> Test-AzureADConnectHealthConnectivity -Role ADFS
Test-AzureADConnectHealthConnectivity's execution in details are as follows:
Starting Test-AzureADConnectHealthConnectivity ...

Connectivity Test Step 1 of 3: Testing dependent service endpoints begins ...
AAD CDN connectivity is skipped.
Connecting to endpoint https://login.microsoftonline.com
Endpoint validation for https://login.microsoftonline.com is Successful.
Connecting to endpoint https://login.windows.net
Endpoint validation for https://login.windows.net is Successful.
Connecting to endpoint https://policykeyservice.dc.ad.msft.net/clientregistrationmanager.svc
Endpoint validation for https://policykeyservice.dc.ad.msft.net/clientregistrationmanager.svc is Successful.
Connecting to endpoint https://policykeyservice.dc.ad.msft.net/policymanager.svc
Endpoint validation for https://policykeyservice.dc.ad.msft.net/policymanager.svc is Successful.
Connectivity Test Step 1 of 3 - Testing dependent service endpoints completed successfully.

Connectivity Test Step 2 of 3 - Blob data upload procedure begins ...
Tenant Id is successfully collected during agent registration.
Connectivity Test Step 2 of 3 - Blob data upload procedure completed successfully.

Connectivity Test Step 3 of 3 - EventHub data upload procedure begins ...
Tenant Id is successfully collected during agent registration.
Connectivity Test Step 3 of 3 - EventHub data upload procedure completed successfully.

Test-AzureADConnectHealthConnectivity completed successfully...
```

Now that we've worked through the Azure AD Connect Health Service, we'll continue our journey with the Azure AD monitoring and logs features.

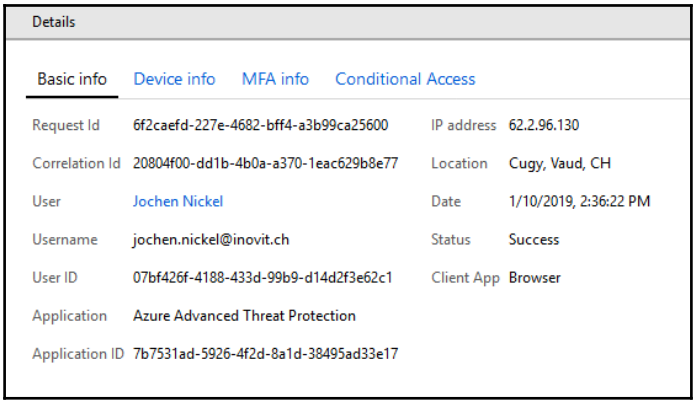
Azure AD monitoring and logs

In the Azure portal, under the Azure AD blade and the **Monitoring** section, we also get insights into the Azure platform and the associated on-premises identity infrastructure.

We can view the complete **Sign-ins** to the several services, as well as **Audit logs**, additional logs, and diagnostic information. The following screenshot shows the actual **Sign-ins** to our environment, including the time, user, application, and status. Furthermore, you will see whether the access was protected by a conditional access rule or Azure MFA itself:

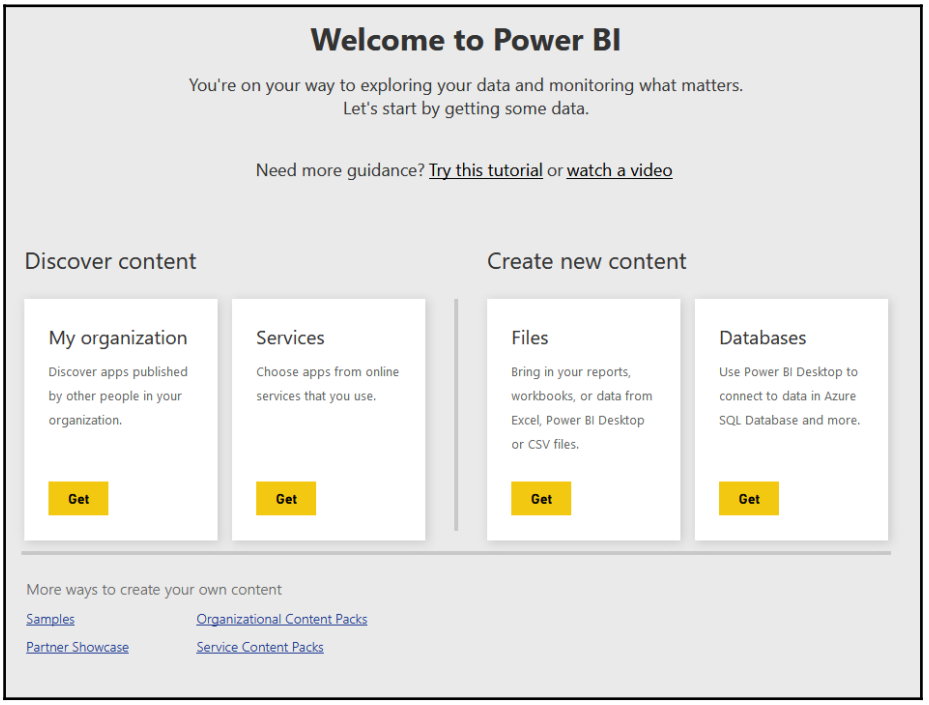


By clicking on an entry, you get many more details:



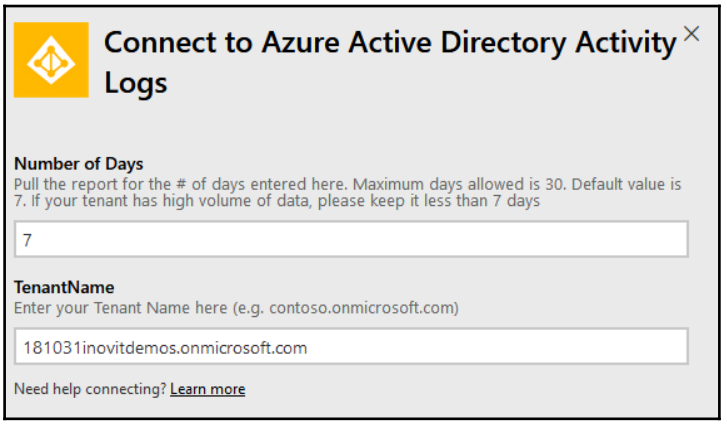
To provide programmatic access to your Azure Active Directory data over the REST-based APIs, you can use this guide: `https://docs.microsoft.com/en-us/azure/active-directory/reports-monitoring/tutorial-access-api-with-certificates`.

You're also able to extend your reporting capabilities with **Power BI** by clicking the **Power BI** icon. If you do so, the following screen appears:



1. Click **My organization** and we'll start to extend our solution with **Power BI**. We can use the predefined apps to produce a good starting point. Choose the **Azure Active Directory Activity Logs** app – just press **Get it now**. You can find more information about this pack at `https://docs.microsoft.com/en-us/azure/active-directory/reports-monitoring/howto-power-bi-content-pack`.

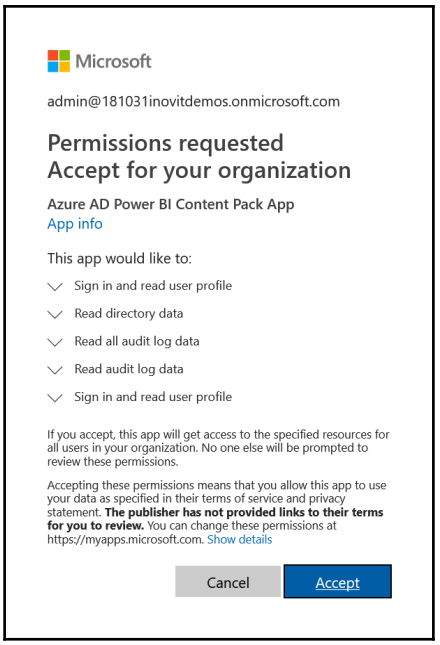
2. Provide the number of days you want to pull for your reports and your tenant name. Click **Next**:



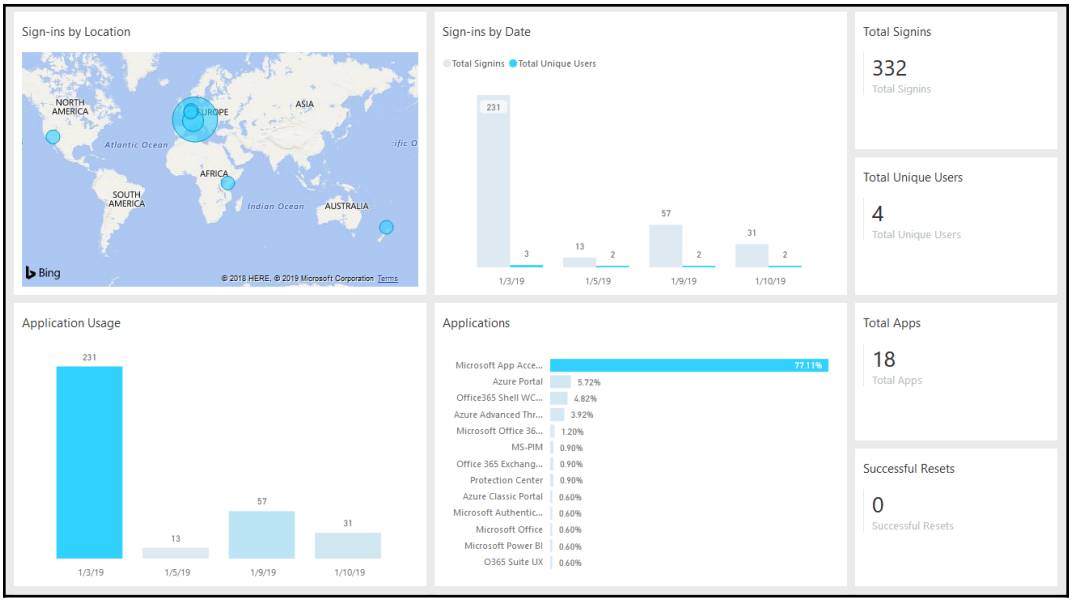
3. The wizard will guide you to connect, and then you need to sign in to consent for `OAuth2`.

We'll discuss the security issues of this process in detail in `Chapter 6`, *Managing Authentication Protocols*.

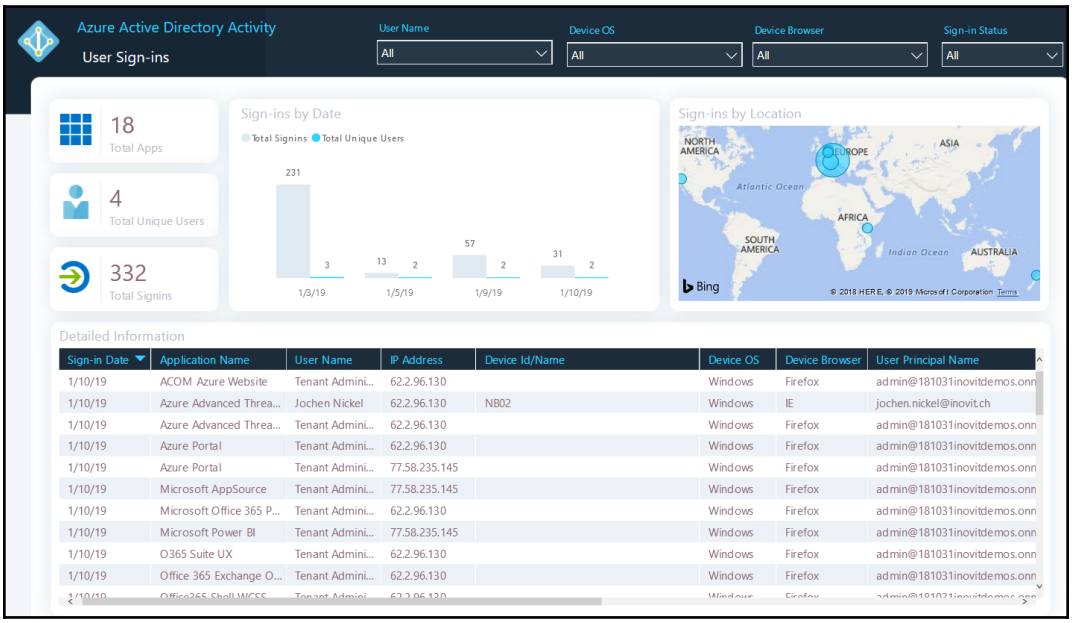
4. Click **Accept**:



That was easy, wasn't it? From now on, you can gather details about your complete identity infrastructure through fancy reports:



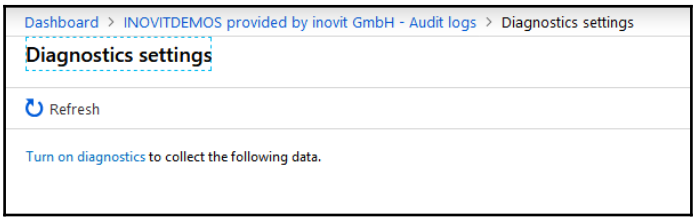
Keep in mind that all information is built in a drill-down format:



Next, we'll take a deeper look into the **Audit Logs,** where we find our administrative tasks and, for example, the consent we have given to the **Azure AD Power BI pack**:

1/10/2019, 8:55:44 PM	ServicePrincipal : Azure AD Power BI Content Pack App	admin@181031inovitdemos.onmicrosoft.com	Consent to application
1/10/2019, 8:55:44 PM	ServicePrincipal : Azure AD Power BI Content Pack App, User :...	admin@181031inovitdemos.onmicrosoft.com	Add app role assignment grant to user
1/10/2019, 8:55:44 PM	ServicePrincipal : Microsoft Graph, ServicePrincipal : b3a0a59...	admin@181031inovitdemos.onmicrosoft.com	Add OAuth2PermissionGrant
1/10/2019, 8:55:44 PM	ServicePrincipal : Windows Azure Active Directory, ServicePri...	admin@181031inovitdemos.onmicrosoft.com	Add OAuth2PermissionGrant
1/10/2019, 8:55:44 PM	ServicePrincipal : Microsoft Graph, ServicePrincipal : https://...	admin@181031inovitdemos.onmicrosoft.com	Add app role assignment to service principal
1/10/2019, 8:55:44 PM	ServicePrincipal : Windows Azure Active Directory, ServicePri...	admin@181031inovitdemos.onmicrosoft.com	Add app role assignment to service principal
1/10/2019, 8:55:43 PM	ServicePrincipal : Azure AD Power BI Content Pack App	admin@181031inovitdemos.onmicrosoft.com	Add service principal

Be aware that you can download the information in a CSV format. This option requires you to turn on the **Diagnostic settings**. Just click on **Turn on diagnostics**:



After enabling the diagnostic feature, you get three options for where you can transfer the information:

- Archive to a storage account, which needs to be created or an existing one will be used additional costs. Routing logs to an Azure storage account enable you to retain it for a longer period.
- Stream to an event hub from which you can create activities that should be done. Routing logs to an Azure event hub allows you to integrate with third-party SIEM tools, such as splunk. This enables you to combine different data, such as the Azure AD activities and other security information, in your SIEM to provide in-depth insights into your environment.
- Send to **Log Analytics** to collect the logs on a central position for your SOC. **Log Analytics** consolidates monitoring data from various sources and provides a query language and analytics engine and provides you with insights into the operation of your applications and resources.

The following types are available:

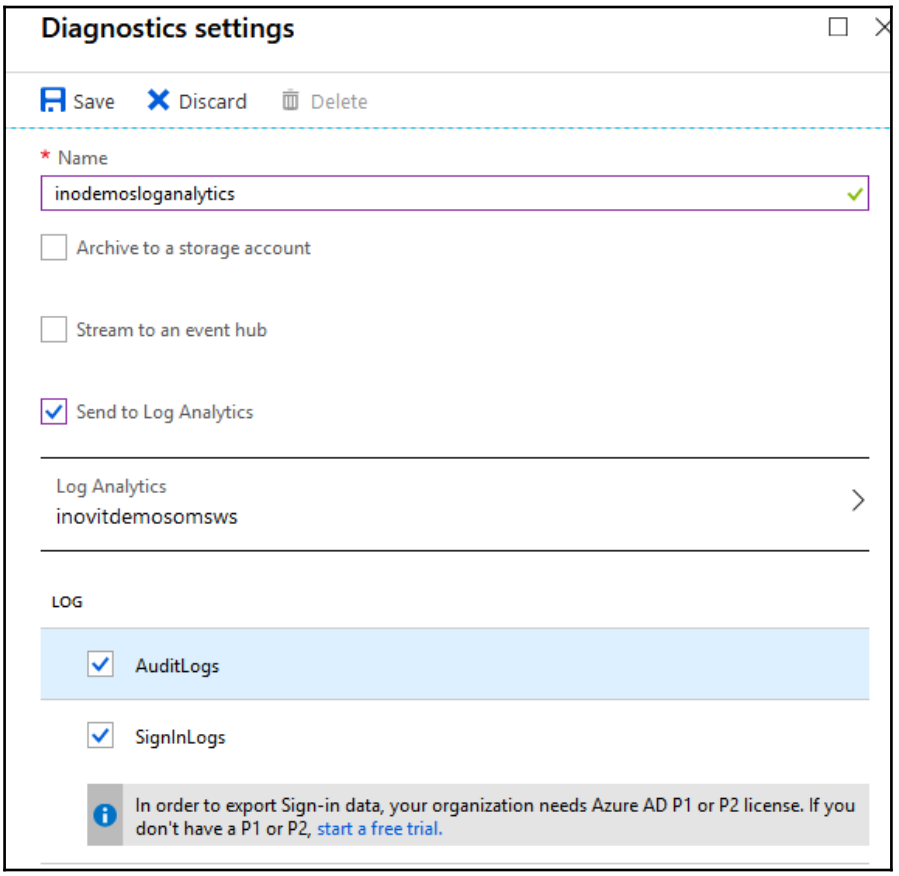
- **AuditLogs,** with information about the operations and activities done by users or services

- **SignInLogs**, with information about the applications used, including conditional access information, login times, and risk levels

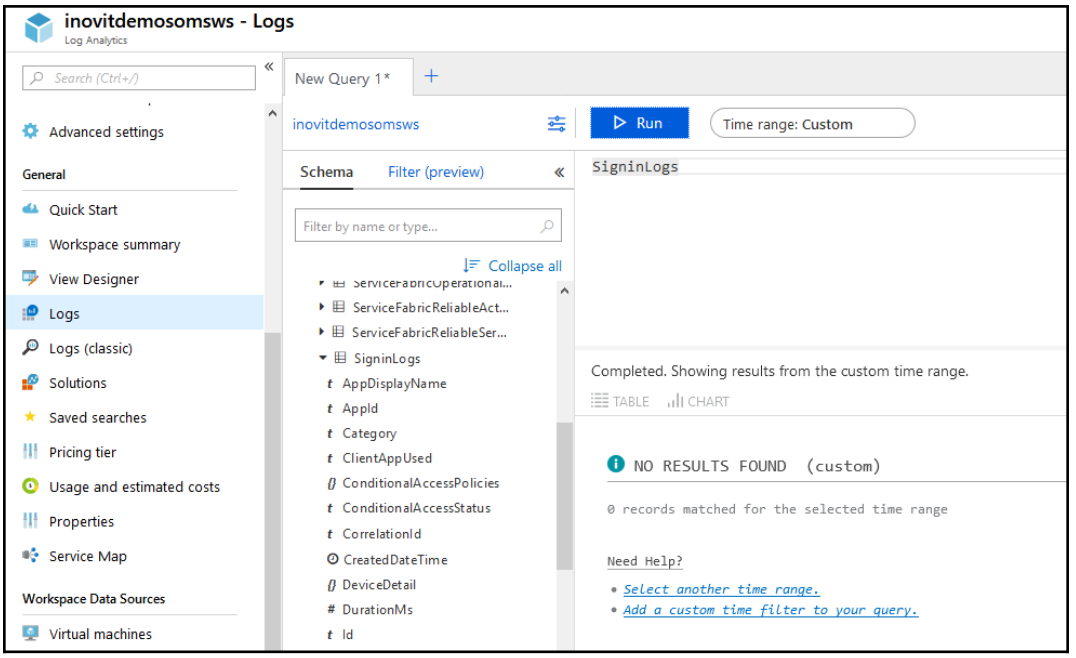
To export sign-in data, your organization needs the Azure AD P1 or P2 license.

We'll turn on the **Log Analytics** functionality with the following information:

- You can add **Name** as `inodemosloganalytics`
- Select **Send to Log Analytics** and check the box for **AuditLogs** and **SignInLogs**:



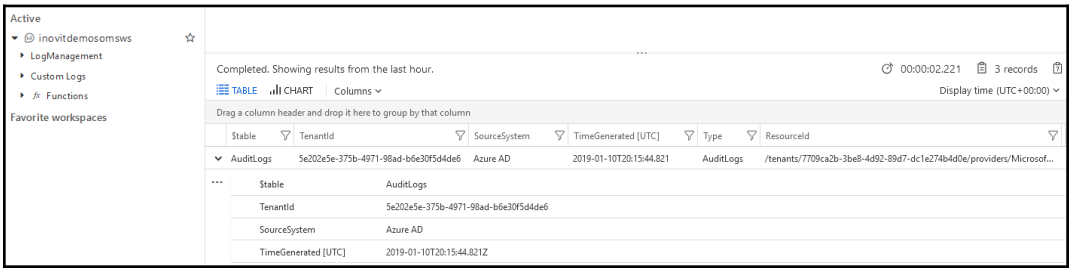
Next, jump to your newly-created log analytics functionality and start querying the information from your Azure AD:



If you want to get more information about this topic, check out the "*Integrate Azure AD logs with* **Log Analytics** *using Azure Monitor*" video at `https://docs.microsoft.com/en-us/azure/active-directory/reports-monitoring/howto-integrate-activity-logs-with-log-analytics`.

Furthermore, we'll use **Log Analytics** in `Chapter 16`, *Azure Information Protection*, in the last part of this book.

If you use the logs part in the Azure AD section, you'll see the first result after the service is activated:



Now that you have an idea of the Azure AD capabilities, you can start trying the different functionality to get a perfect overview of your infrastructure.

Azure Security Center for monitoring and analytics

In this section, we'll explore the Azure Security Center's capabilities for monitoring our identity and access-management infrastructure. With the Azure Security Center, you're able to understand the security state across your on-premises environments and cloud workloads. New Azure resources will be automatically discovered and on-boarded and you can apply security policies across your complete hybrid environment to ensure compliance with actual security standards. The service also provides the collection, search, and analysis of several sources, including third-party solutions and firewalls. Furthermore, we can find threats with advanced analysis mechanisms and are able to respond and recover from incidents because of the provided real-time security alerts. The export of your security events into an SIEM solution for further analysis is possible. The service integrates many new threat-detection parts, such as behavioral profiling, machine learning, and global intelligence threats.

The Azure Security Center can find many different attacks, such as the following:

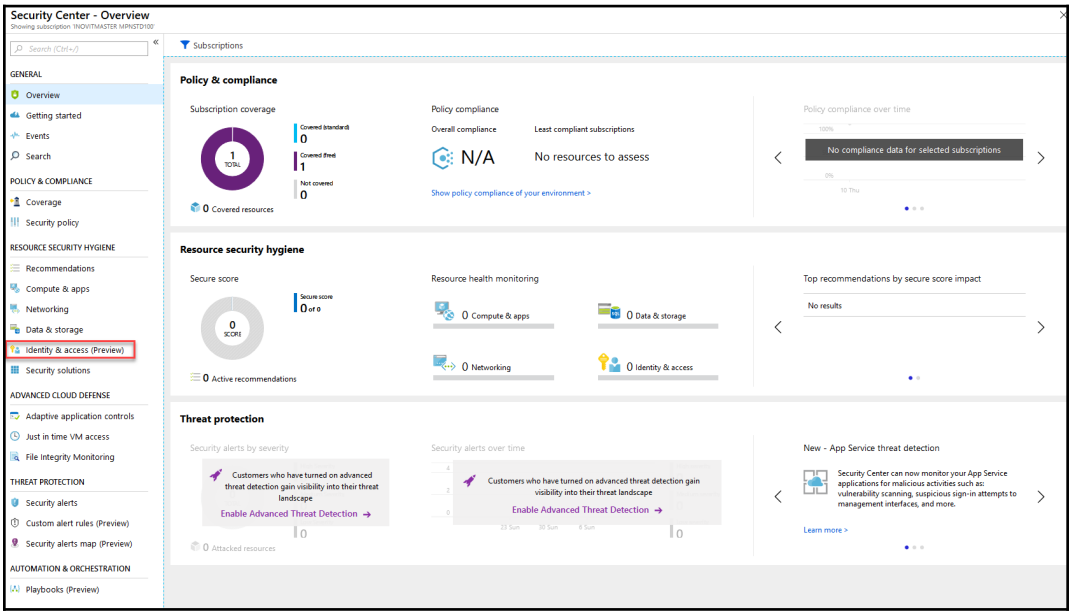
- Compromised machines
- Failed exploitation attempts
- Advanced malware
- Web application vulnerabilities
- Brute-force attacks
- Data exfiltration

The monitoring identity and access feature set is in preview. You need to use a standard-tier plan of the Azure Security Center to use this functionality. Actually, there is a limit of 600 accounts for using the identity recommendations.

With the monitoring of identity activities, you can provide a higher level of security for your organization, because you can provide proactive actions before an incident happens. Furthermore, you are in the position to stop an active attack. The identity and access dashboard provides you with insights and recommendations for your subscription, such as the following:

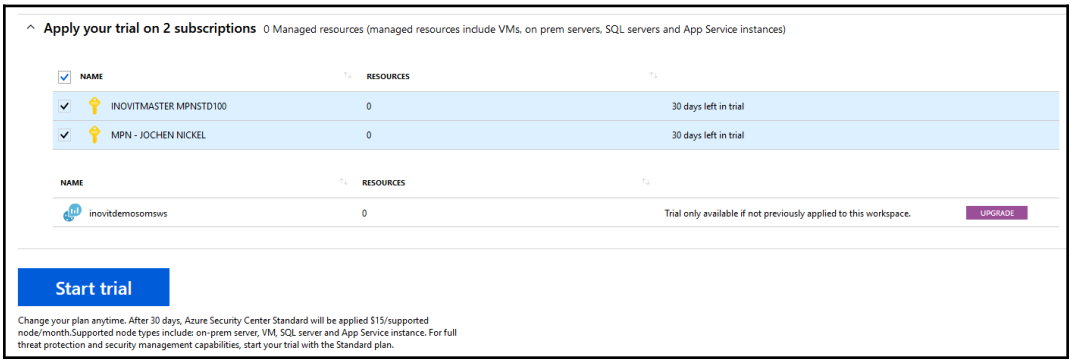
- Enable MFA for privileged accounts
- Remove external accounts with write permissions
- Remove privileged external accounts

We'll find the following message if we start on the portal: **You have subscriptions that are not fully protected. Upgrade to the Standard tier for more recommendations**:



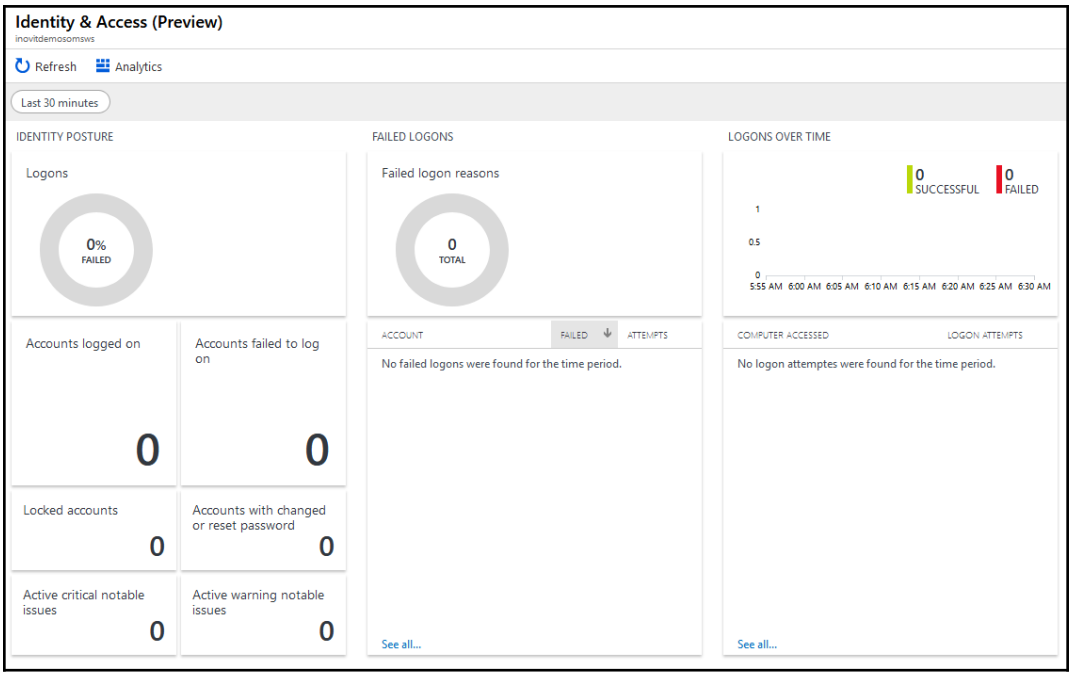
With the following steps, we will enable the feature set:

1. Upgrade to an Azure Security Center standard instance:



2. Start a trial and upgrade your existing log analytics workspace, with the following expected result:

3. You can see the insights into your identity infrastructure, where you can define different time frames to be analyzed:



We also see that we're directly able to dive deeper into the monitoring and log information through the log-analytics integration, which can be used on the top of the portal. We highly recommend that you come back to the monitoring portals during the work on all labs provided in this book to see the several components in action.

Summary

In this chapter, we discussed the key monitoring and log capabilities to hold your identity infrastructure in a stable and suitable state. We learned which functionality is provided by the Azure AD section in the Azure portal. Furthermore, we saw how Azure AD Connect Health works and how easy it is to install and configure your environment. We extended our solution and now have a broader view of the whole security state of the hybrid environment through the Azure Security Center. You should be able to answer questions about the identity infrastructure-monitoring capabilities and to configure a suitable monitoring and log solution for your own organization or customers.

In the next chapter, we will learn how to use the Azure Identity Protection features to protect your cloud and on-premises environment. We will discuss and configure the Azure AD Privileged Identity Management capabilities.

5
Configuring and Managing Identity Protection

After reading about and learning how to configure the monitoring of your identity management infrastructure in the previous chapter, we will now dive into identity protection. Protecting your identity is one of the main focuses of security today, so you should be able to put the right capabilities in place that protect your organization against any attack.

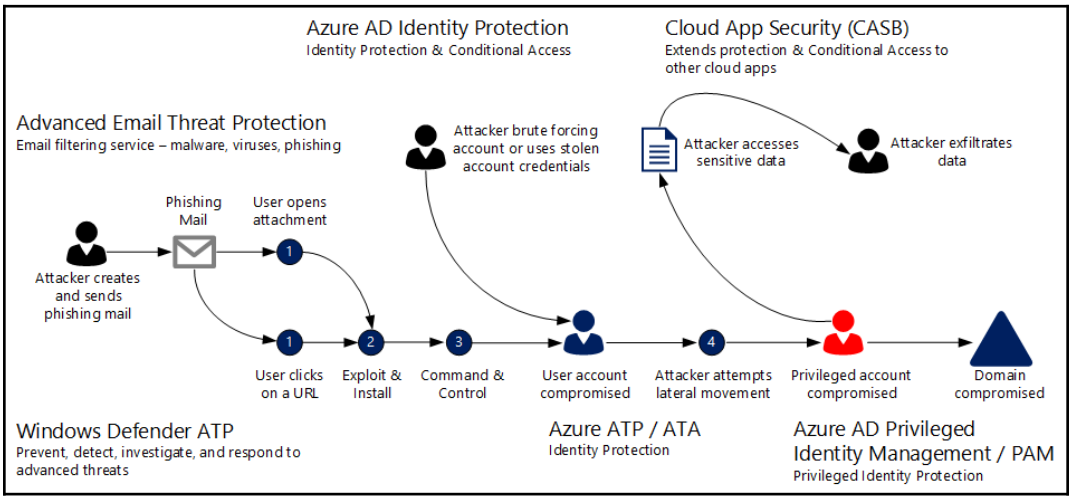
At the beginning of our identity protection journey, we will start with an overview of the Microsoft Cloud services that can help you in this field. We will also dive into a number of different services, starting with the theory before moving on and applying that theory in an example configuration. After working through this chapter, you should be able to identify the correct solution component for your existing or future requirements. To summarize, this chapter covers the following topics:

- Microsoft Identity Protection solutions
- Azure ATP and how to use it
- Azure AD Identity Protection
- Using Azure AD **Privileged Identity Management** (**PIM**) to protect administrative privileges

Let's start with an introduction to Microsoft Identity Protection solutions so we can maximize problem detection during different attack stages.

Microsoft Identity Protection solutions

Microsoft has developed and established a huge portfolio of identity protection features on both Azure and on-premises to help organizations secure their identities and protect themselves against sensitive data leakage. There are many single services available, but a unified identity protection solution is expected in the near future. We already see many integrations between identity protection services that detect, investigate, and prevent advanced attacks, compromised identities, and insider threats—including data leakage—to form a broad and deep identity investigation solution that works on-premises and in the cloud. Some of the more common attacks that are faced by organizations include password spray attacks, leaked or reused credentials, spoofed domains, malicious links or attachments, and much more. The following figure illustrates the different services often used to maximize the detection of attacks in their various stages:



Attack vectors and protection products overview

As you can see in the above figure, an attacker first successfully targets patient zero with backdoor malware designed to establish command and control. The attacker then moves laterally through the network and obtains privileged credentials and accesses sensitive systems. During this stage, the Office 365 **Advanced Email Threat Protection**, **Windows Defender ATP**, the **Azure ATP**, and **Azure AD Identity Protection** should all be in place to protect the identities at stake. An attacker will then try to use remote code execution from a local machine to a Domain Controller in order to gain access to domain administrator accounts.

It's important at this stage that **Azure ATP** and **Azure AD Privileged Identity Management** are used to establish the right controls. After this step, if successful, the attacker is free to access sensitive data and open it exfiltrate such information. To protect your sensitive data, the **Microsoft Cloud App Security Broker** (**MSCASB**) and the **Cloud App Security** service can help you to extend the protection of your sensitive data.

Now that we've had a look at a potential attack scenario and the related technology involved, we can dive deeper into each of the related services. We'll use the **crawl-walk-run** strategy to start, as follows:

- **Crawl**: First, protect your privileged users and primary environments by:
    - Enabling Azure MFA for your administrative accounts, or better, using Azure AD PIM
    - Monitoring your risk reports and using the identity secure store functionality
    - Enabling and configuring Azure ATP to protect your primary user domains
- **Walk**: Then, protect all users and Domain Controllers by:
    - Blocking legacy authentication and using conditional access (for more information, visit: `https://bit.ly/2M3NXzw`)
    - Turning on password hash synchronization in Azure AD Connect and banned password checking (for more information, visit: `https://docs.microsoft.com/en-us/azure/active-directory/authentication/concept-password-ban-bad`)
    - Protecting all domains and forests using Azure ATP
    - Monitoring all Azure ATP alerts and investigating any lateral movements or domain dominance alerts
- **Run**: Finally, protect all users and integrate the alerts into your security operations flow by:
    - Enabling Azure MFA for your end users using sign-in and the user risk policy

In this chapter, we will focus on identity protection technologies that are mentioned in the preceding strategy. Other processes will be looked at in `Chapter 13`, *Identifying and Detecting Sensitive Data* and `Chapter 8`, *Using the Azure AD App Proxy and the Web Application Proxy*. However, now, let's start with the Azure ATP service.

Azure ATP and how to use it

Azure ATP is used to detect and investigate advanced attacks, compromised identities, and insider threats. Thanks to behavioral analytics in the backend, it provides very fast threat detection and also reduces the fatigue of false positives. Furthermore, it provides focused essential information using the Azure ATP attack timeline. Azure ATP is simple to work with, and the architecture is quite easy to understand because there are only two components per service and a downloadable sensor, which monitors local traffic, that is installed directly on your Domain Controllers. The sensors use dynamic resource limitation based on the domain controller's load.

There is another, more complex deployment method available, however, which uses a standalone sensor on a dedicated server and requires the configuration of port-mirroring from the Domain Controllers in order to receive network traffic. The service integrates directly with the Microsoft Intelligent Security Graph. You can find more information about this functionality at `https://www.microsoft.com/en-us/security/operations/intelligence`.

Azure ATP helps you to detect the following advanced attacks:

- Reconnaissance attacks, such as:
    - Account enumeration
    - User group membership enumeration
    - User and IP enumeration
    - Host and server name enumeration (DNS)
- Compromised credentials, such as:
    - Brute force attempts
    - Suspicious VPN connections
    - Suspicious group membership modifications
    - Honey Token account suspicious activity
- Lateral movements, such as:
    - Pass-the-Ticket
    - Pass-the-Hash
    - Overpass-the-Hash

- Domain dominance, such as:
    - Golden Ticket attacks
    - DC shadowing
    - Skeleton Key attacks
    - Remote code execution on Domain Controllers
    - Service creation on Domain Controllers

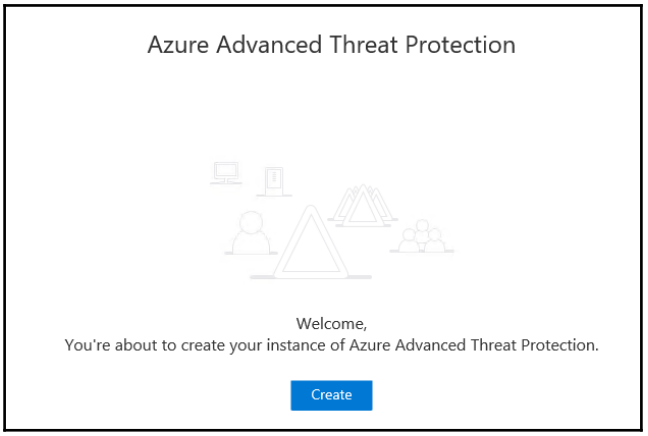
The on-premises product with the most basic features available is Microsoft **Advanced Threat Analytics** (**ATA**), which is still supported and in development. The cloud service, on the other hand, provides the functionality to protect your environment, so we recommend Azure ATP over ATA.

We will now see how to configure Azure ATP in your environment.

You need to have finished `Chapter 1`, B*uilding and Managing Azure Active Directory* and `Chapter 2`, *Understanding Identity Synchronization* to complete the configuration which follows.

So, let us get started! To configure Azure ATP, you need to follow the below steps:

1. Open the link, `https://portal.atp.azure.com/`, in a browser and log on as a global administrator.
2. Next, create your instance of **Azure Advanced Threat Protection**, as shown in the following screenshot:

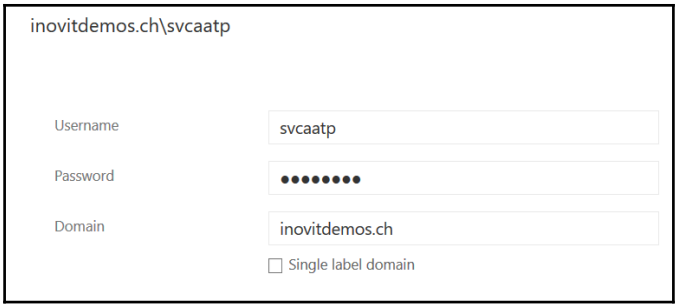


Azure ATP creation dialog

3. Before configuring Azure ATP, you need to create a service account in your local active directory. Log on to your Domain Controller and open an elevated PowerShell and execute the following cmdlet:

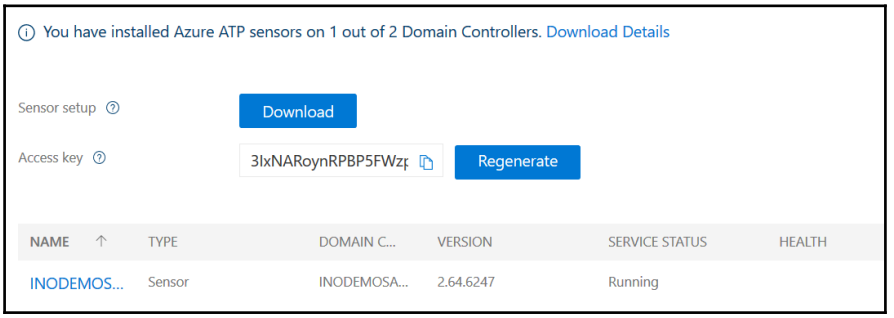
```
New-ADUser -Name "Azure ATP Service Account" -SamAccountName
svcaatp -UserPrincipalName svcaatp@inovitdemos.ch -path
"OU=Users,OU=AAD,OU=Managed Service
Objects,DC=inovitdemos,DC=ch" -AccountPassword (ConvertTo-
SecureString "YourPassword" -AsPlainText -Force) -Enabled
$True
```

4. Jump back to the Azure AD portal and follow the configuration instructions.
5. Now, use the newly-created service account, as shown in the following screenshot:



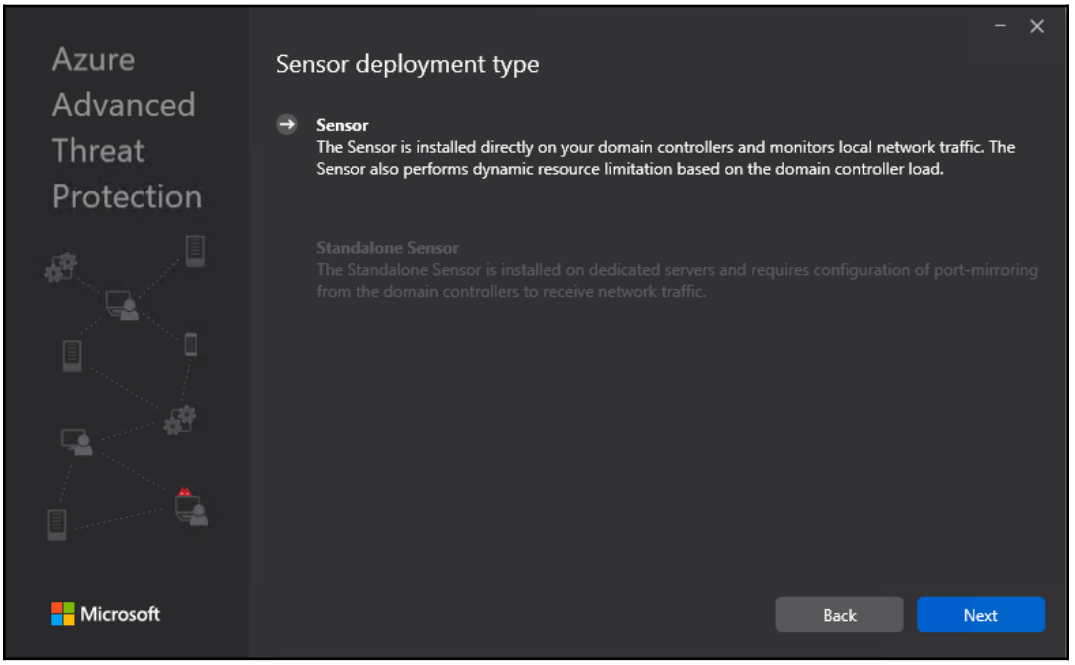
Connecting the directory services

6. Afterward, directly deploy your first Azure ATP sensor on your **Domain Controller**, as shown in the following screenshot:



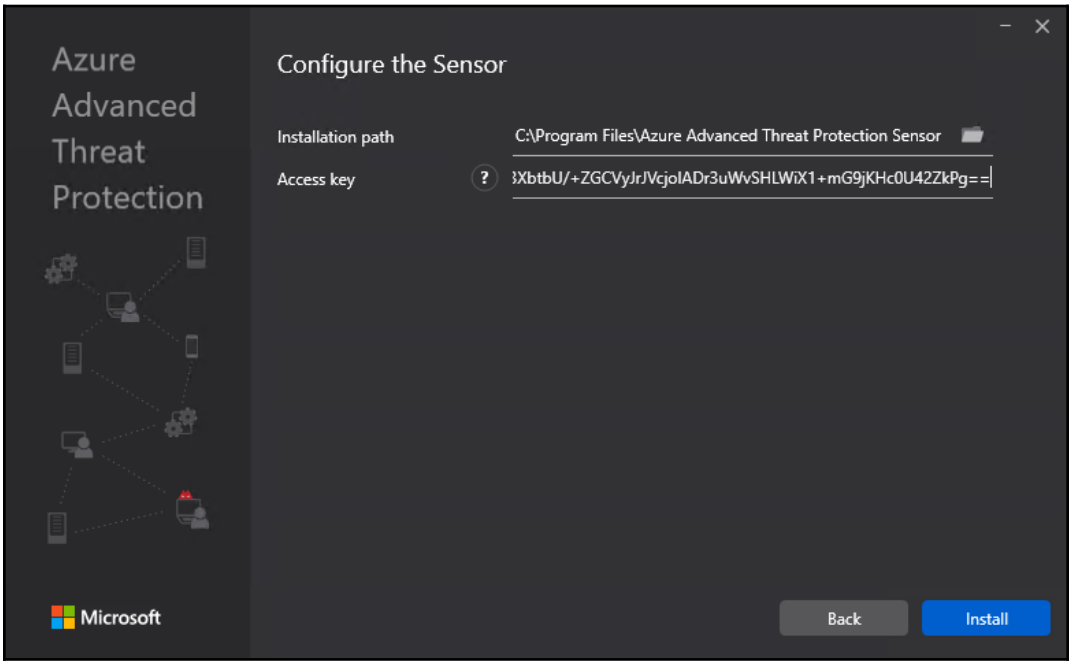
Deploying the Azure ATP agents

7. Download the setup executable to your Domain Controller and start the installation.
8. Choose your preferred language and choose the **Sensor**-only deployment method, as shown in the following screenshot:



Choosing the sensor type

9. Click **Next** and **Configure the Sensor** with the **Access key** of the portal, as shown in the following screenshot:

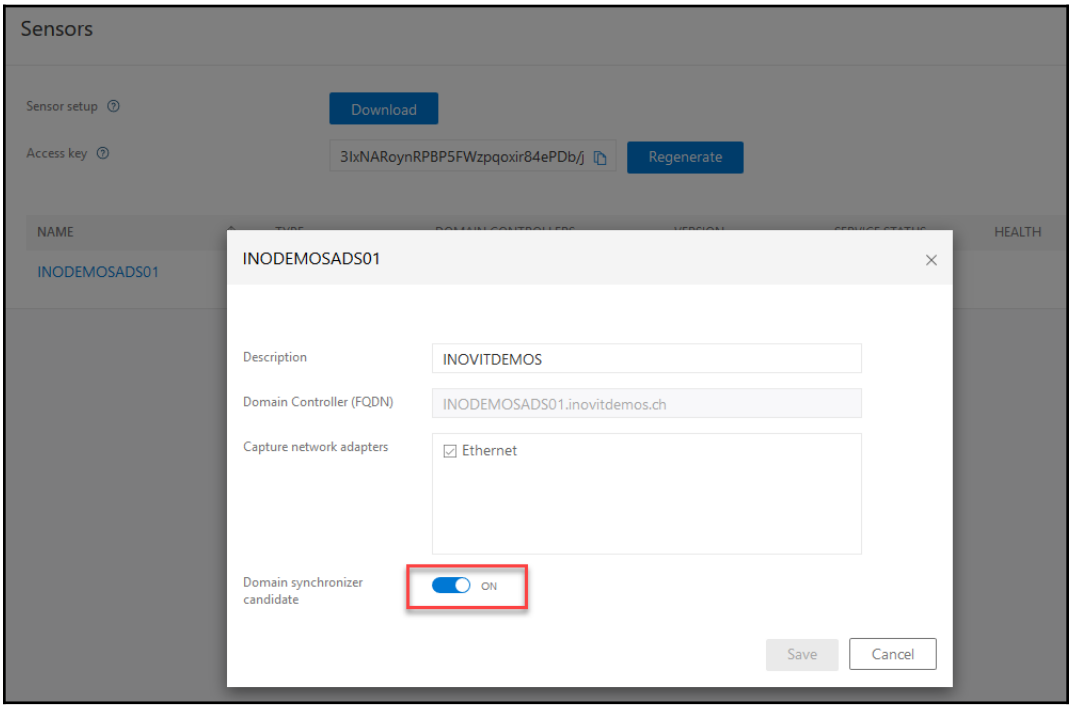


Getting the sensor agent shared key and installation path

10. Finish the installation and open the `services.msc` management console to view the newly-installed services, which should include the following two services:

Newly installed services for Azure ATP

11. Next, enable the sensor as a **Domain synchronizer** to gather the related information for Active Directory, as shown in the following screenshot:

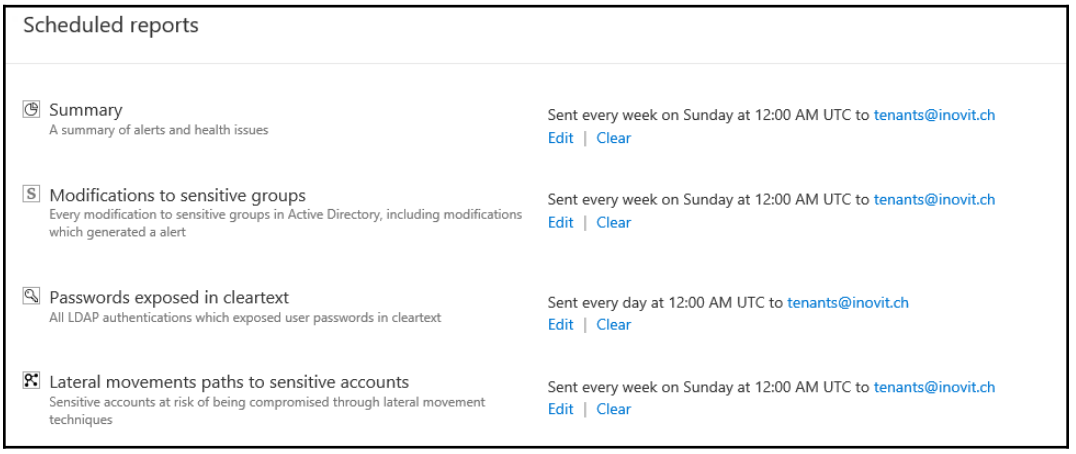


Enabling the synchronizer role

12. The next configuration task is the activation of the **Suspected Golden Ticket usage** detection, which can be found under the **Preview** section, as shown in the following screenshot:

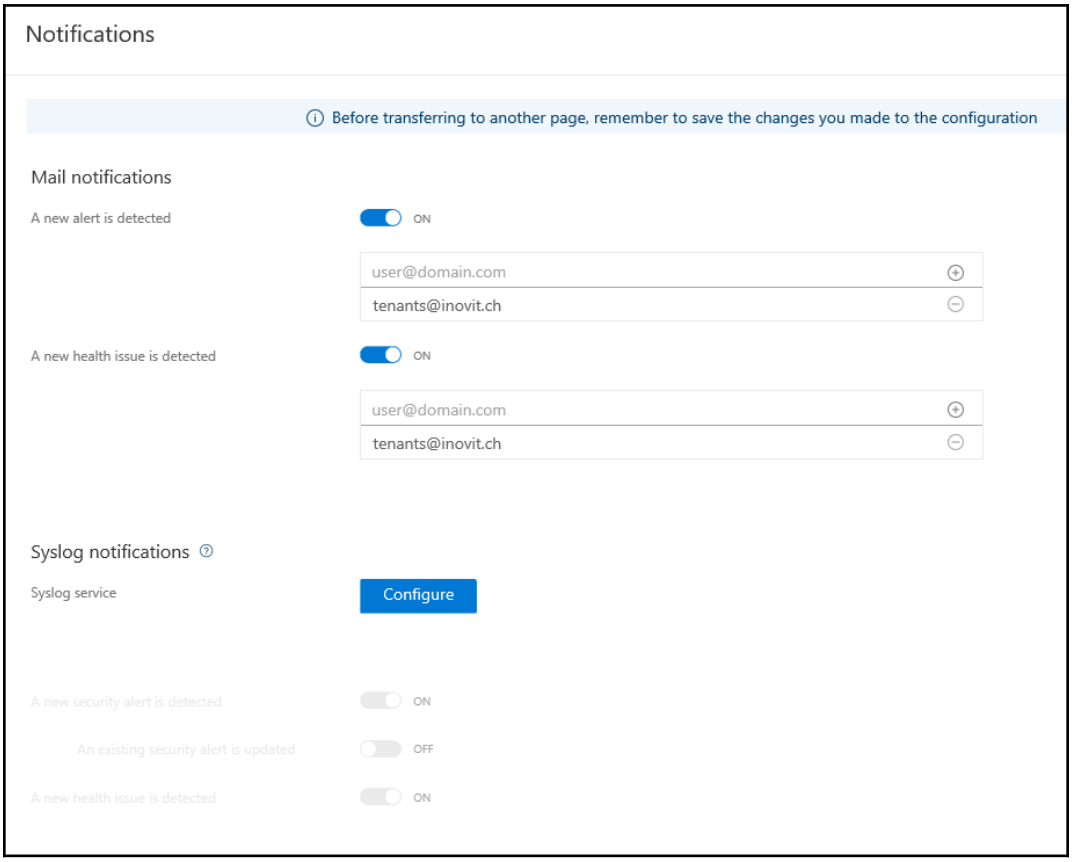
Activation of the golden ticket protection

13. Configure all of the relevant **Scheduled reports** and notification settings for testing its capability, as shown in the following screenshot:



Reporting section overview

14. The **Notifications** settings page looks like the following screenshot:



Configuring notifications

Now that you have configured your Azure ATP service, you can use the guidance available at `https://gallery.technet.microsoft.com/ATA-Playbook-ef0a8e38` to create attacks and test your environment in realistic scenarios. The document is written for Microsoft ATA, but you can still use it for testing purposes. You can also use the following Azure ATP security alert guide to validate your results: `https://docs.microsoft.com/en-us/azure-advanced-threat-protection/suspicious-activity-guide`. In the next section, we will dive into the features of Azure AD Identity Protection.

Azure AD Identity Protection

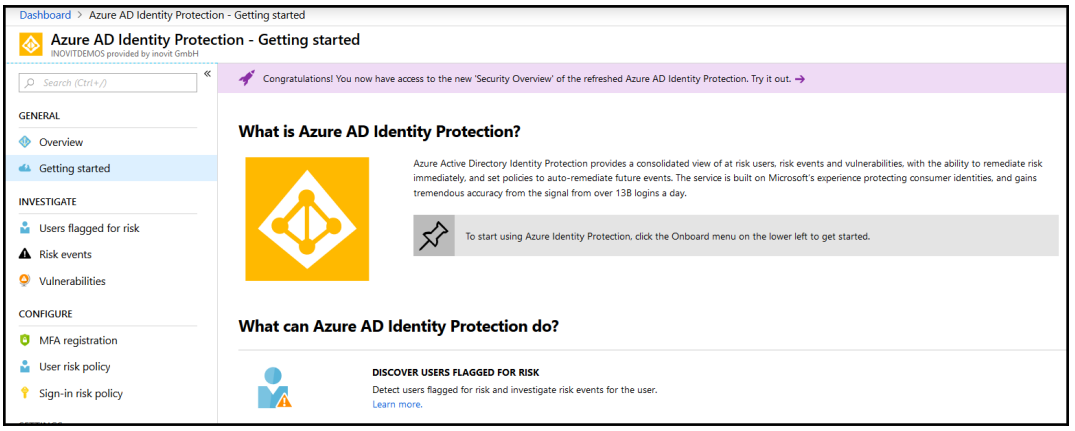
Azure AD Identity Protection introduces automatic, risk-based, conditional access to help protect users against suspicious logins and compromised credentials. Azure AD Identity Protections also offers insight into, and a consolidated view of, threat detection based on machine-learning. Furthermore, the service delivers an important level of remediation recommendations, as well as performing compromise risk calculations about a user and their session. The service requires an Azure AD Premium P2 or equivalent licensing.

You will get the following capabilities from this service:

- **Detection**: Vulnerabilities and risky accounts are detected by:
    - Highlighting vulnerabilities and providing custom recommendations
    - Calculating sign-in and user risk levels
- **Investigation**: Risk events are investigated and solved by:
    - Notifications
    - The provision of relevant and contextual information
    - Basic workflows used in tracking
    - Easy access to remediation actions (for example, a password reset)
- **Risk-based conditional access**: It includes:
    - Risk mitigation, such as blocking sign-ins or requesting multi-factor authentication
    - Blocking or securing risky user accounts
    - Asking users to register for Azure MFA

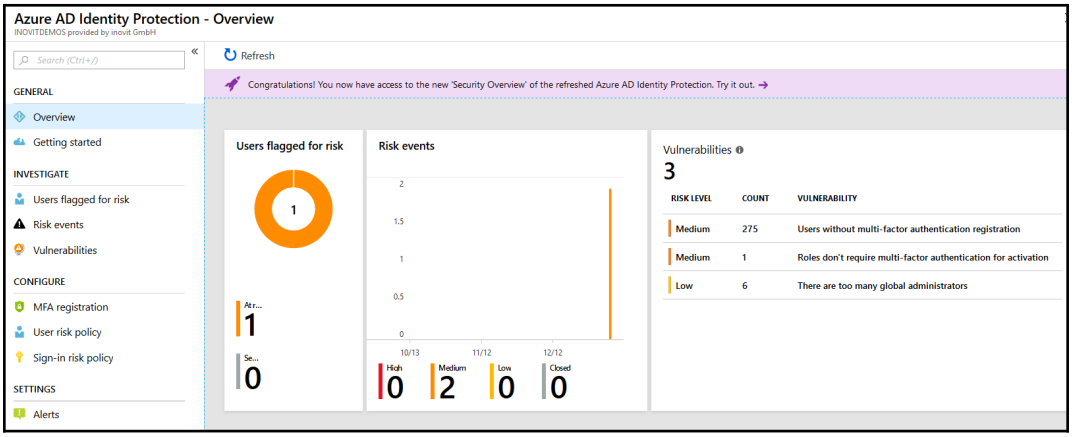
To get a greater understanding of this service, we need to start configuring it. So, login as a global administrator to `https://portal.azure.com` and complete the following steps:

1. Go the **Azure AD Identity Protection** tab or use the **Search** option to search the service
2. Start the on-boarding process by visiting the onboard menu under **Settings** and clicking **Create** at the bottom of the page, as shown in the following screenshot. Follow the process and finish on-boarding:



Azure AD Identity Protection wizard

3. Now jump directly into the **Overview** section, as shown in the following screenshot:

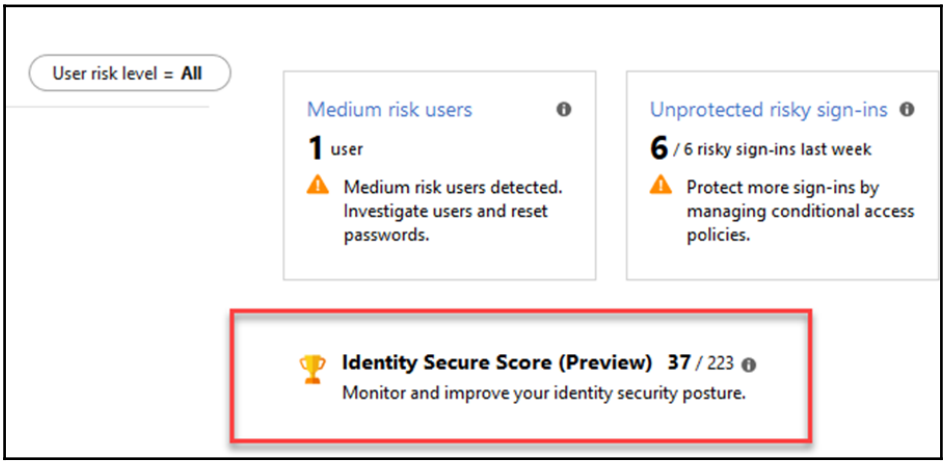


Azure AD Identity Protection portal overview

4. The portal will present you with the relevant risks facing your environment, such as:
    - Users who are not enrolled with Azure MFA
    - Azure AD or Azure resource roles that don't require Azure MFA
    - The assignment of too many global administrator roles

5. Now, configure your first scenario by using the following source `https://docs.microsoft.com/en-us/azure/active-directory/identity-protection/quickstart-sign-in-risk-policy`, with one of your users
6. Next, configure the alerts and weekly digest settings for information about the activities in and against your environment
7. After the service is enabled, you can test the functionality of Azure AD Identity Protection with the existing playbook, which is available at: `https://docs.microsoft.com/en-us/azure/active-directory/identity-protection/playbook`.

As you can see, running Azure AD Identity Protection doesn't require some long configuration task. However, we do recommend that you run your tests based on the provided playbook and get as much experience as possible:



Identity Secure Score information

As shown in the preceding screenshot, the **Identity Secure Score** provides you with information on your identity security position and how it can be improved. The service will recommend activating certain security features and following specific recommendations, for example:

- Enabling Azure MFA for Azure AD privileged roles
- Enabling self-service password reset
- Enabling sign-in and user risk policies
- Not allowing the expiration of passwords, or turning on password hash sync if you are in a hybrid configuration
- Ensuring all users are enrolled to Azure MFA
- Not using more than five global administrators
- Disabling accounts that have not been used in the last 30 days
- Blocking legacy authentication
- Not allowing users to grant consent to unmanaged applications

We will discuss some areas of identity protection in more detail in `Chapter 6`, *Managing Authentication Protocols* and `Chapter 13`, *Identifying and Detecting Sensitive Data*. We will also offer a deep-dive into the Cloud App Security service, which provides you with capabilities to protect against the following:

- **Malicious insiders**: Protection against disgruntled employees
- **Malware**: Detection of malware in cloud storage as soon as they are uploaded
- **Rogue applications**: Identification of rogue applications that can access your sensitive data
- **Compromised accounts**: Combating advanced attackers that leverage compromised user credentials
- **Data exfiltration**: Detection of unusual flows of data either within or outside of your organization
- **Ransomware**: Identification of ransomware that uses sophisticated behavioral analytics technology

Now that we have looked at Azure AD Identity Protection, it's time to move on to the Azure AD PIM service.

Using Azure AD PIM to protect administrative privileges

Azure Active Directory **Privileged Identity Management** (**PIM**) provides similar functionality to the Microsoft Identity Manager, including **Privileged Access Management** (**PAM**) in the on-premises infrastructure.

If you need more information about the on-premises PAM solution, visit `https://docs.microsoft.com/en-us/microsoft-identity-manager/pam/privileged-identity-management-for-active-directory-domain-services`.

With Azure AD PIM, you can manage, control, and monitor your privileged identities and access to your directory information and resources in an Azure environment. The main reason for using Azure AD PIM is to reduce the attack surface and to enable administrative access just-in-time. Privileged access is often configured as permanent and unmonitored, but with Azure AD PIM you can avoid security breaches and risks.

The service allows you to assign time-bound access to resources using a start and end date and that requires approval to activate privileged roles. To protect the activation of a role, the service uses Azure Multi-Factor Authentication. For example, during the activation process, a user can be forced to justify why they need to activate their role. Furthermore, you can also enable notifications that alert you when a privileged role is activated. For auditing and compliance requirements, you are also able to configure and enable access reviews that ensure a user needs a specific role. You can also download an audit history for both internal and external audits.

To use Azure AD Privileged Identity Management, you need to own an Azure AD Premium P2 license, which is also part of the EMS E5 or Microsoft 365 E5 plan, for every privileged account you want to protect. If you don't have access to an Azure AD Premium P2 license, protect your permanent privileged accounts with Multi-Factor Authentication. To do so, you will need to meet the licensing requirements, which are explained in full at `https://docs.microsoft.com/en-us/azure/active-directory/authentication/concept-mfa-licensing`.

Now that we know the basic functionally of Azure AD PIM, let's start with an example configuration.

To work through this lab, you must have completed the lab tasks in the first chapters, or have your own users already created.

The following table illustrates a common example configuration used by many companies when they first begin using the cloud:

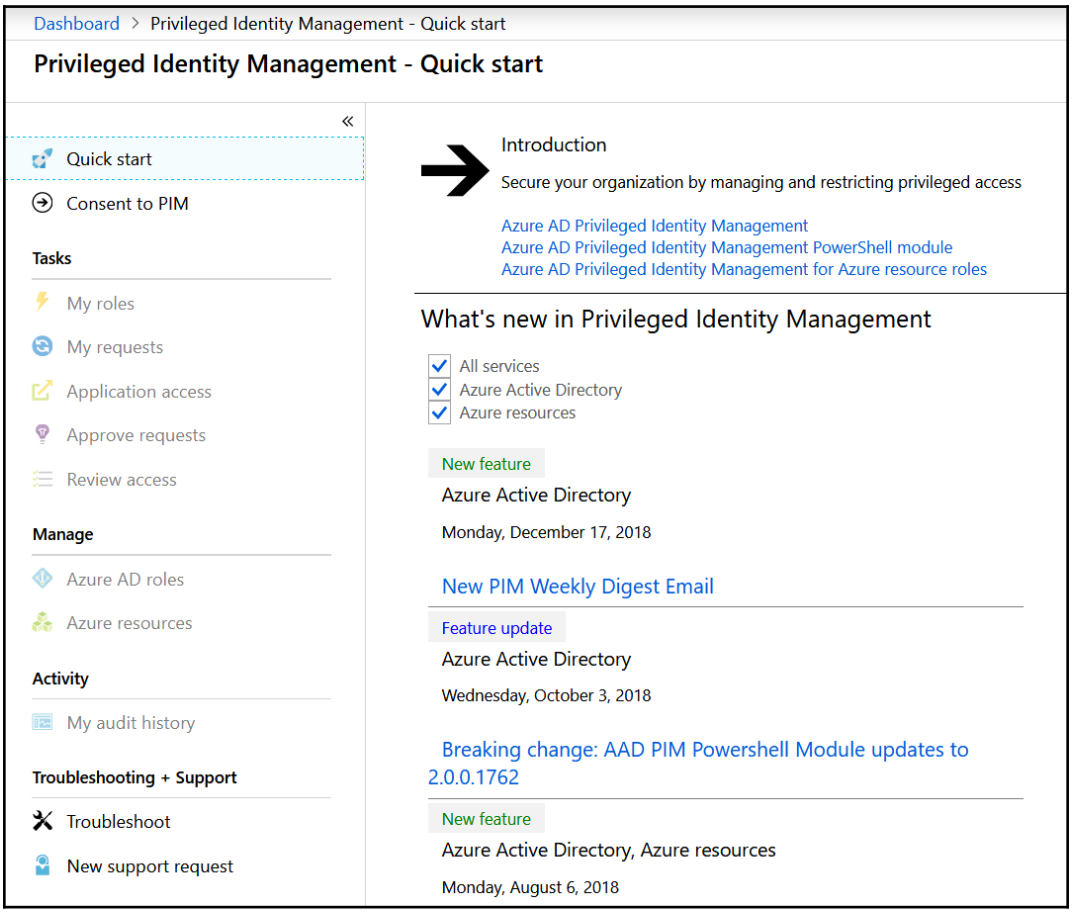
Directory role	User
Global Administrator	`Don.Hall@inovitdemos.ch`
Global Administrator	`Dan.Jump@inovitdemos.ch`
Global Administrator	`Chris.Gray@inovitdemos.ch`

Keep in mind that unmanaged privileged accounts create a vulnerability that can be exploited to access critical systems with sensitive data. Continuously randomizing credentials for each account—whether scheduled or in direct response to an attack—is the first step to take.

First, let's enable Azure AD PIM on our test tenant and use the default role configurations. We need to ensure that it works and to remember to implement a breaking-glass account. Afterward, we will be able to gather feedback and adjust our configuration as necessary.

Now we will start the configuration by enabling PIM with the Quick Start wizard.

1. First, log in with your global administrator account at `https://portal.azure.com` and navigate to Azure AD PIM.

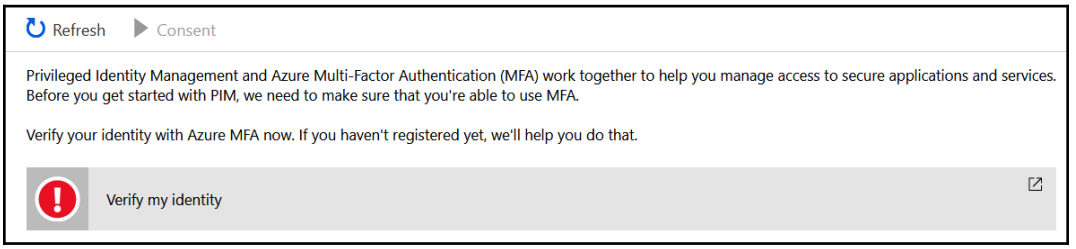


Azure AD PIM - Quick Start options

In our case, we use the account `admin@inovitdemos.onmicrosoft.com`, which will act as our breaking-glass account:

A breaking-glass account allows you to modify your configuration even if Azure AD PIM, the Multi-Factor Authentication service, or the federation service, is not available. Make sure to protect this account and use your Azure AD PIM controlled account to administer your environment.

2. Now enable the service by using **Consent to PIM**, as shown in the preceding screenshot.
3. Now verify your identity, as shown in the following screenshot:



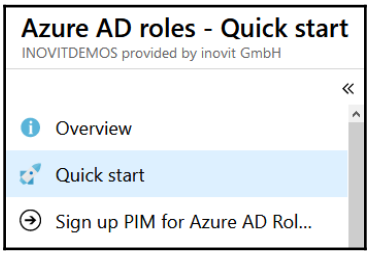
Identity verification process

The more information required dialog box appears if Azure MFA hasn't been previously enabled.

4. Fill in and finish the additional security verification dialog.
5. Click **Consent** to proceed, as shown in the following screenshot:

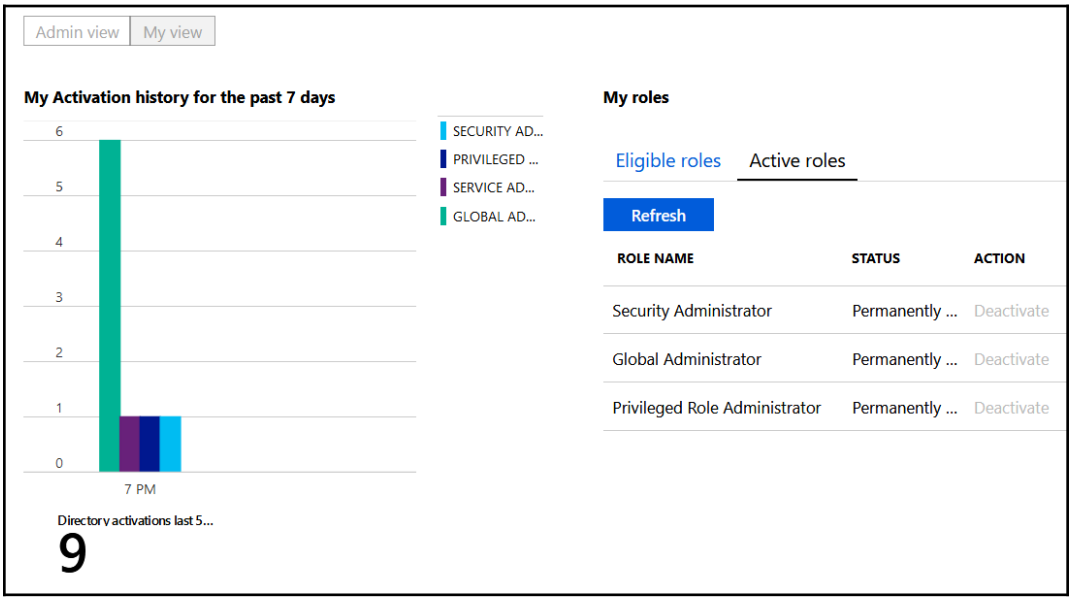
Consent to activate the feature

6. Next, select **Sign up PIM for Azure AD Roles** under the **Manage** | **Azure AD roles** section, as shown in the following screenshot:



Sign up process to PIM

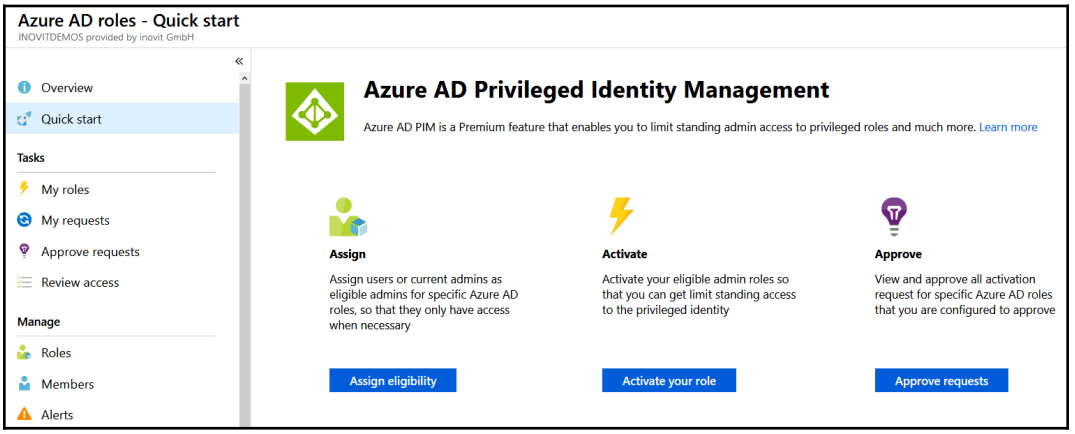
7. Click **Sign up** to start the discovery process for administrative roles.
8. View your own **Active roles** in the portal, as shown in the following screenshot, including their status:



Insides about the active roles

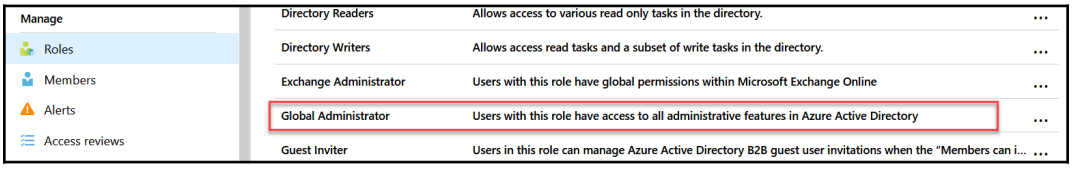
In the next tasks we will add users to roles and test the configuration.

1. Now, return to **Azure AD roles** page and manually assign the three global administrators as eligible admins for the specific Azure AD, as shown in the following screenshot. Note that you can also use the **Manage | Wizard** option to modify your role assignments:



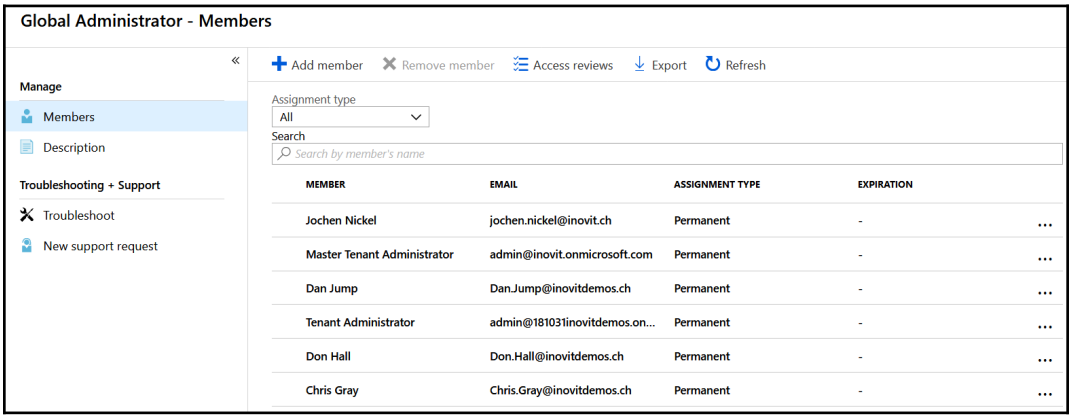
Modification of role assignments

2. Click on **Manage | Roles** and choose the **Global Administrator** role, as shown in the following screenshot:



Configuring the Global Administrator role

3. You should see all of the members present, including their assignment type, as shown in the following screenshot:

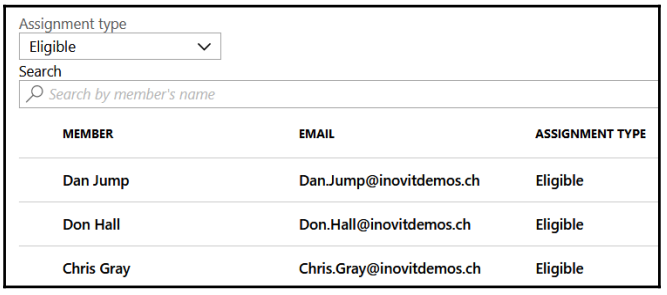


Assigned roles overview

4. Now edit your three test users so they are eligible by clicking the three dots at the end of the user line, as shown in the following screenshot:

Choosing the assignment type to make him eligible

5. You should end up with the following result:



Result of changes

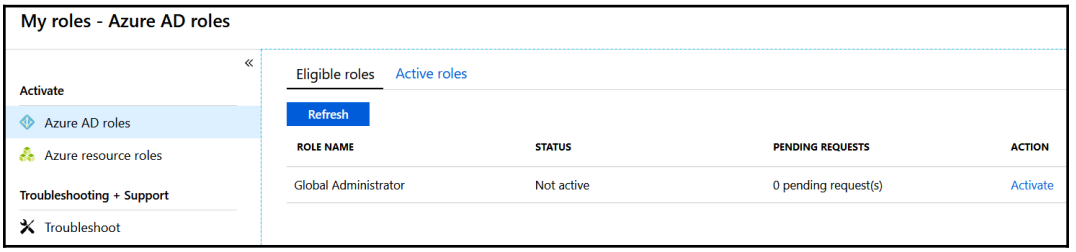
Practical note: Check the description of the role and see what you can do with it.

6. Click on **Access reviews** to create an access review for a role.
7. Click **New.**
8. Choose and fill in the following values because it's a best practice to start with a monthly review:
    - **Review name**: Global Administrator role monthly review
    - **Description**: Global Administrator role monthly review
    - **Frequency**: Monthly
    - **Review role membership**: Global Administrator
    - **Reviewers**: Selected users
    - **Select reviewers**: Use your `admin@tenantname.onmicrosoft.com` (PIM role administrator)
    - **Upon completion settings**:
        - **Auto apply**: Enable
        - **Reviewer not responding**: Remove access
9. Leave the remaining default settings and then click **Start**. You should end up with the following result:

Access review overview

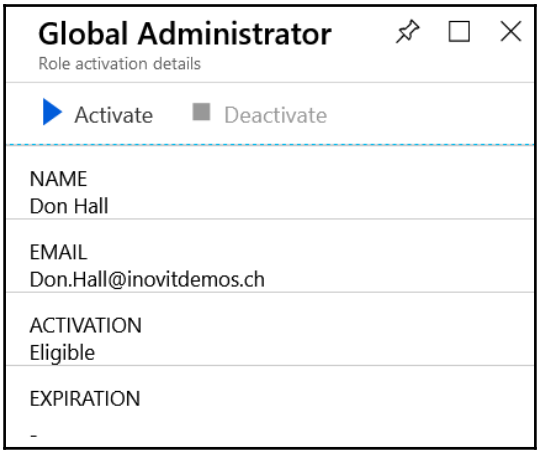
10. Now that we have converted the accounts from permanent to eligible and have created our first access review for the **Global Administrator** role, we will test our configuration with Don Hall.
11. Open a new private browser session and navigate to `https://portal.azure.com`.
12. Log in with `don.hall@yourdomain.com`.
13. Navigate to Azure AD PIM or use the **Search** option to find it.

14. Click on **My roles**. You should find that you are eligible for the **Global Administrator** role, as shown in the following screenshot:



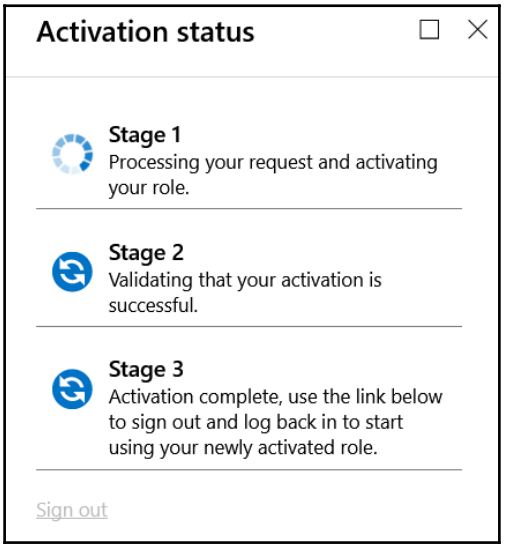
View your own roles

15. Click **Activate** to use your role. You will need to verify your account before proceeding.
16. Click **Verify my identity** to complete Azure MFA enrollment.
17. Respond to the verification process and click **Activate**, as shown in the following screenshot:



Activate the global administrator role

18. Note that you can customize activation time; the role is active for 1 hour by default. Now, provide an activation reason and click **Activate**. Then, sign out and try your first login as G**lobal Administrator**:



Proofing the activation

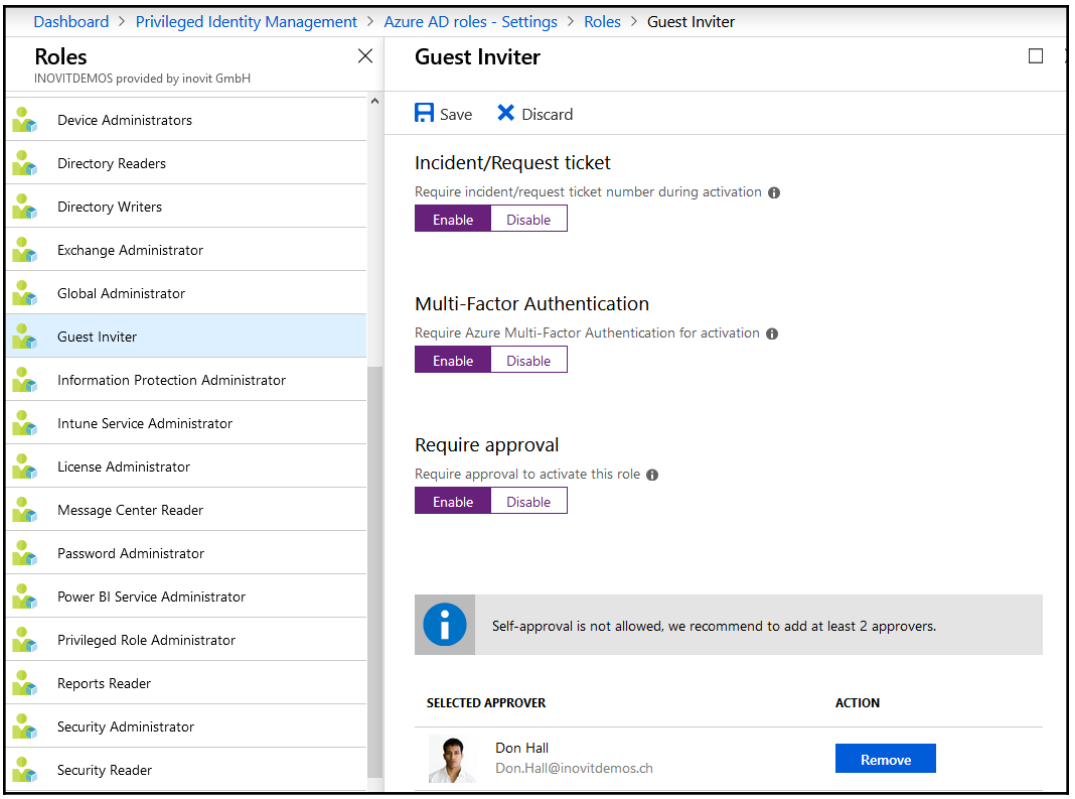
19. That's it! You've just successfully gained privileged access with Azure AD PIM.
20. Now, sign Don Hall out and log back in with your permanent administrator that also has the privileged administrator role assigned.

21. Click on **Activity** | **Directory roles audit history** to find the role activation from Don Hall, as shown in the following screenshot:

Roles audit history

Check the following features on the **Azure AD roles** section to get more experience on how to configure Azure AD PIM:

- Go to **Tasks** | **Application Access** to verify your role assignment.
- Go to **Review access** to complete a review.
- Go to **Approve requests** if you have configured a role approval; you can do this under **Manage** | **Settings** | **Roles**:



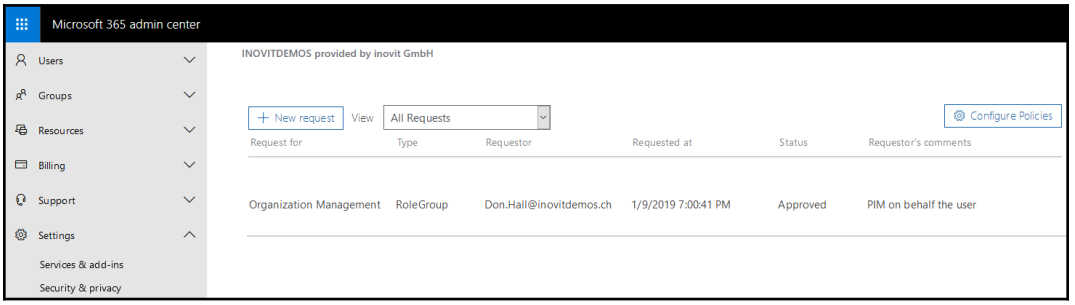
Guest Inviter role assignment

Now that we have successfully enabled Azure AD PIM for **Azure AD roles**, assigned and tested our first role, created an access review for the role, and viewed the audit history, we're closer to understanding Azure AD **Privileged Identity Management**. (Note that there is an additional lab for Azure resources and other features in the code section of the book for this chapter.)

To summarize, you should complete the following Azure AD PIM tasks for your Azure resources:

- Enable Just in Time access to Azure
- Expire access automatically
- Assign temporary access for quick tasks or on-call schedules
- Get alerts when new users or groups are assigned resource access, or when eligible assignments are activated
- Use Azure AD sign in for Azure VMs

Before we close our section on PIM, we should tell you that Azure AD PIM, and Office 365 E5, offer Privileged Access Management functionality for privileged tasks and workload roles:



Office 365 Privileged Identity Management portal

Now that we have configured the Privileged Identity Management and the protection, we provide a good state of security to our organization.

Summary

After completing this chapter, you should be able to explain the main requirements when protecting identities and why they're a part of your security solution. In this chapter, we looked at the key issues of core identity protection components, and how to enable and configure the relevant services for your needs. If you would like more information on the Windows Defender ATP service, check out the `Chapters 13`, *Identifying and Detecting Sensitive Data*.

In the next `Chapter 6`, *Managing Authentication Protocols*, we will discuss the all-important modern protocols, including OAuth 2.0, OpenID Connect, and SAML 2.0, which help you to establish a suitable authentication design for your organization and customers.

2
Section 2: Authentication and Application Publishing

This section of the book will enable you to understand and use as well as troubleshoot with and develop on the most important authentication protocols. Furthermore, you will learn to securely publish cloud and on-premise services with Azure AD and ADFS/Web Application Proxy capabilities.

The following chapters will be covered in this section:

- `Chapter 6`, Managing Authentication Protocols
- `Chapter 7`, Deploying Solutions on Azure AD and ADFS
- `Chapter 8`, Using the Azure AD App Proxy and the Web Application Proxy
- `Chapter 9`, Deploying Additional Applications on Azure AD
- `Chapter 10`, Exploring Azure AD Identity Services
- `Chapter 11`, Creating Identity Life Cycle Management in Azure

6
Managing Authentication Protocols

In this chapter, we will provide you with an overview about the important authentication protocols you need to know in order to handle your configurations and projects in that field.

We see a lot of confusion in the usage of authentication protocols in our projects. It's very important to understand the different protocols, that you can discuss with application providers about the correct implementation tasks and requirements. We see very often that a lot of time is used for discussing authentication methods and solutions. It's clearly impossible to put all the material about the different authentication methods in only one chapter, as that would fill a complete book in the real world. We decided to provide you with essential summaries with extensive and working external examples. We will deploy many of the different authentication methods in the labs of this book. You will find specific labs to adapt your knowledge of technical configurations in the following chapters:

- `Chapter 7`, *Deploying Solutions on Azure AD and ADFS*
- `Chapter 8`, *Using the Azure AD App Proxy and the Web Application Proxy*
- `Chapter 9`, *Deploying additional applications on Azure AD*
- `Chapter 10`, *Exploring Azure AD identity services*

The focus of this chapter will be to get you acquainted with some experiences and decision paths that can assist you with authentication protocols. We will divide the chapter into the following sections:

- Microsoft identity platform
- Common token standards in a federated world
- **Security Assertion Markup Language** (**SAML**) 2.0
- WS-Federation
- OAuth 2.0
- **OpenID Connect** (**OIDC**)
- Pass-through authentication and seamless **single sign-on** (**SSO**)
- **Multi-factor authentication** (**MFA**)

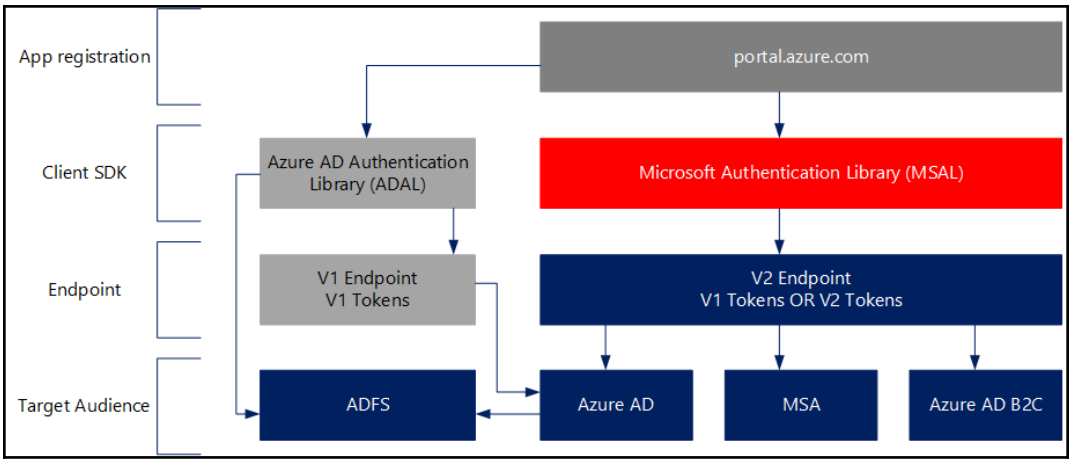
Let's start working through this authentication reference to kick off your journey.

We highly recommend that you work through all the references in this chapter.

We will start with an overview of the Microsoft identity platform.

Microsoft identity platform

Microsoft provides an identity platform with two endpoints called V1.0 and V2.0 with two sets of client libraries to work with these endpoints. The following libraries are used: Azure **AD Authentication Library** (**ADAL**) SDK and the **Microsoft Authentication Library** (**MSAL**). In the Azure AD portal, we will find how to include applications built with ADAL or MSAL over the app registrations (preview), as shown in the following screenshot:



Microsoft identity platform overview

The following list describes the main use cases of the two endpoints:

- The V1.0 endpoint allows only work and school accounts to sign in
- The V2.0 endpoint allows work and school accounts from Azure AD and **Microsoft Accounts** (**MSA**) to sign in
- The v2.0 endpoint does not support SAML or WS-Federation—only OIDC and OAuth 2.0
- The v2.0 endpoint doesn't support the SAML assertion grant
- Both endpoints accept sign-ins from guest users for single-tenant or multi-tenant applications

Now that we have an overview on the available endpoints, we will jump into the token formats.

Common token standards in a federated world

When a digital identity is transferred across a network, it's only a collection of bytes. It's common to refer to a set of bytes containing identity information as a security token or just a token. In a claims-based world, a token contains one or more claims, each of which carries some piece of information about the user it identifies.

Today, the tokens come in different flavors, including the following token formats:

- **Security Assertion Markup Language** (**SAML**):
  - XML-based
  - Very descriptive metadata
- **JSON Web Token** (**JWT**):
  - Easily readable by humans
  - Smaller token size
- **Simple Web Token** (**SWT**):
  - Form-encoded attribute/value pairs
  - Not very common
- **Kerberos**

For the following protocol specifications, we recommend a good basic knowledge of claims-based authentication. You can download the Microsoft claims-based authentication handbook to prepare yourself. Use the following download link to get the book: `https://www.microsoft.com/en-us/download/details.aspx?id=28362`.

We will discuss SAML 2.0 in the next section.

Security Assertion Markup Language (SAML) 2.0

SAML is the foundation for much of the current identity federation activity. SAML 2.0 is preceded by SAML 1.0 and 1.1. SAML 1.1 was released in 2003 and had just two scenarios (also known as profiles), and both were IdP-initiated. Shibboleth 1.3 and Liberty Alliance—WS-FF 1.2 extended SAML 1.1, and SAML 2.0 was released by OASIS in 2005.

The following table shows the SAML core principles:

Assertions	**Protocols**	**Bindings**
Package of identity information	Request/response based	Associates a message (protocol) with transport (communication mechanism)
Synonym token	Defines the messaging requirements	Examples: • HTTP Redirect • HTTP POST • HTTP Artifact • SOAP
XML-based	Examples: • Authentication request • Single logout • Artifact resolution	

In the next section, we will talk about the key facts of the SAML 2.0 protocol.

Key facts about SAML

The SAML standard provides accurate messages for the transfer of requests and assertions (claims). SAML offers several options for the transfer of information, such as the use of SOAP. The SAML standard defines identity information as assertions. A large part of the standard flows into the definition of assertions and attribute profiles. A session timeout is not considered. When logging off, an attempt is made to reach a large circle of recipients. SAML 2.0 uses profiles, which describe how assertions, protocols, and bindings combine to form a federation scenario.

For example, a web SSO profile will be used as follows:

- Authentication Request Protocol
- HTTP Redirect binding at IdP
- HTTP POST Binding at SP

There are many different SSO profiles available that are defined in the specification:

- Web Browser SSO profile
- **Enhanced Client or Proxy** (**ECP**) profile
- Identity provider discovery profile
- Single logout profile
- Name identifier management profile

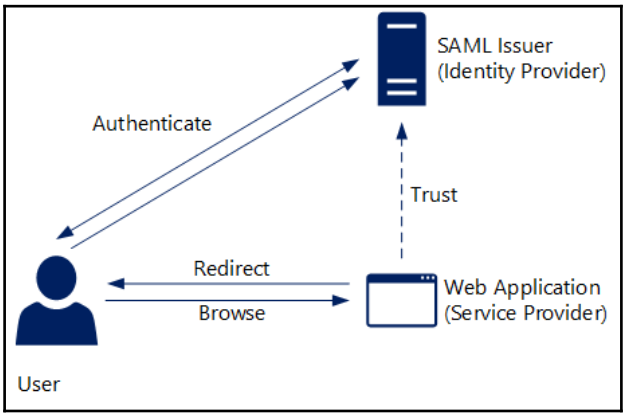
- Artifact resolution profile
- Assertion query/request profile
- Name identifier mapping profile

For Shibboleth administrators, we highly recommend the following source: `https://bit.ly/2DnG5pQ`.

There are also SAML attribute profiles, such as the following, available for use:

- Basic attribute profile
- X.500/LDAP attribute profile
- UUID attribute profile
- DCE PAC attribute profile
- XACML attribute profile

To give an example, we use the typical SAML web SSO profile in the following diagram:



SAML web SSO profile

The following description explains the different steps in the authentication process:

- Trust is established between the **Web Application** and the **SAML Issuer**
- The user browses to the **Web Application**
- The **Web Application** detects that the user is not authenticated and redirects him to the **SAML Issuer**

- The user automatically browses to the **SAML Issuer**
- The user authenticates to the **SAML Issuer**
- The **SAML Issuer** builds the token and passes it back to the user
- The user POSTs the token to the **Web Application**

We highly recommend you work through the following Azure AD SAML 2.0 article: `https://docs.microsoft.com/en-us/azure/active-directory/develop/single-sign-on-saml-protocol`.

For debugging SAML-based SSO, work through the following articles:

- `https://social.technet.microsoft.com/wiki/contents/articles/31247.azure-active-directory-how-to-debug-saml-based-single-sign-on-to-applications.aspx` - How to debug SAML-based single sign-on to applications
- `https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-v1-debug-saml-sso-issues` - Debug SAML-based single sign-on to applications in Azure Active Directory

In conclusion, we can say that SAML 2.0, which was released in 2005 by OASIS, is now commonly used in web sign-in scenarios, particularly for web applications with an XML token format.

WS-Federation

WS-Federation was developed by an industry consortium and was released in December 2006, with Microsoft being a key contributor. WS-Federation is also part of a larger framework, WS-Security, and builds on the work of WS-Trust from February 2005, defining the following two key principles:

- The protocol for requesting/receiving security tokens
- How trust should be brokered between parties using an **Security Token Service** (**STS**)

It also defines two profiles:

- Active Requestor Profile
- Passive Requestor Profile

WS-* Federation Suite consists of:

- WS-Trust
- WS-Federation
- WS-Policy

In the next section, we will describe the key elements of the WS-Federation specification.

Key facts about WS-Federation

In **WS-Federation**, in contrast to SAML, the token can be anything. Basically no defined messages are used. On the other hand, a suggestion is made for the use of a web service. The WS-Federation standard uses SOAP and makes the tunneling of SOAP available via the Web browser. The token for this standard is not subject to any concrete specification. WS-Federation can use a security token that is a SAML assertion. This means that WS-Federation can also use components from the SAML standard. As with SAML, there is no session timeout and an application must explicitly register to receive the information about a logoff.

There are also some relationships regarding SAML-P with the WS-Federation and the passive requestor profile, for example:

- Similar to **SAML WebSSO** profile
- The following parts are incompatible:
    - Different request and response messages
    - No IdP-initiated use case
    - No Assertion Query profile

Use the following source to gather more information about the topic:

```
https://docs.microsoft.com/en-us/azure/active-directory/develop/azure-ad-federation-metadata
```

With the following sample, you can test and analyze WS-Federation with Azure AD:

```
https://github.com/Azure-Samples/active-directory-dotnet-webapp-wsfederation
```

Conclusively, we can say that WS-Federation, which was established in 2006, is now commonly used for web sign-in scenarios, and .NET web applications. The token format is agnostic, like SAML, JWT, and so on.

OAuth 2.0

In simple words, authentication is the act of proving who you are, whereas authorization is the act of determining what you can do. OAuth 2.0 is about delegated authorization and not about authentication. It is not a protocol, it's an authorization framework defined in the RFC 6749, *The OAuth 2.0 Authorization Framework*. This can be confusing because there are many cases in which you use OAuth 2.0 to log in to a client web application.

The authentication process must end by figuring out and validating the identity of the end user, but OAuth doesn't do that. OAuth provides time-based tokens, which can be used to access a resource on behalf of the end user without providing any identity information about the end user.

OAuth 2.0 is the existing standard for API security and is a major breakthrough in identity delegation.

Key facts about OAuth 2.0

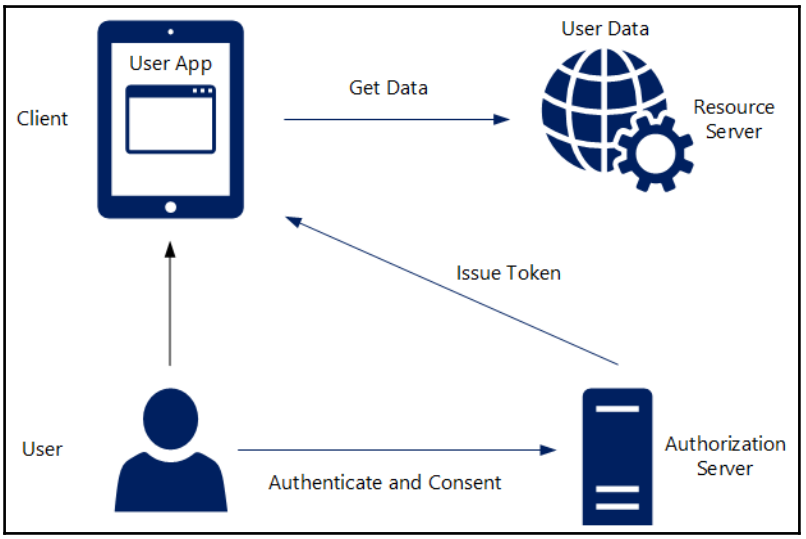
The following are the principal facts concerning OAuth 2.0:

- It is an internet protocol/specification for creating and managing application identity
- It is a cross-platform mechanism
- It has delegated authorization to APIs
- Its main purpose is to get the client an access token
- It is not an authentication protocol
- It is preceded by OAuth 1.0 and OAuth **Web Resource Authorization Profiles** (**WRAP**)
- It is an internet standard used by Facebook, Google, and Twitter

The OAuth framework differs between two client (application) types when accessing a service on behalf of a user. The two types of application can be described as follows:

- **Public**: Runs locally on a device. Not trusted to hold a secret.
- **Private**: Runs behind firewalls. Can be trusted with secrets.

The following diagram gives an example of the roles in the OAuth framework :

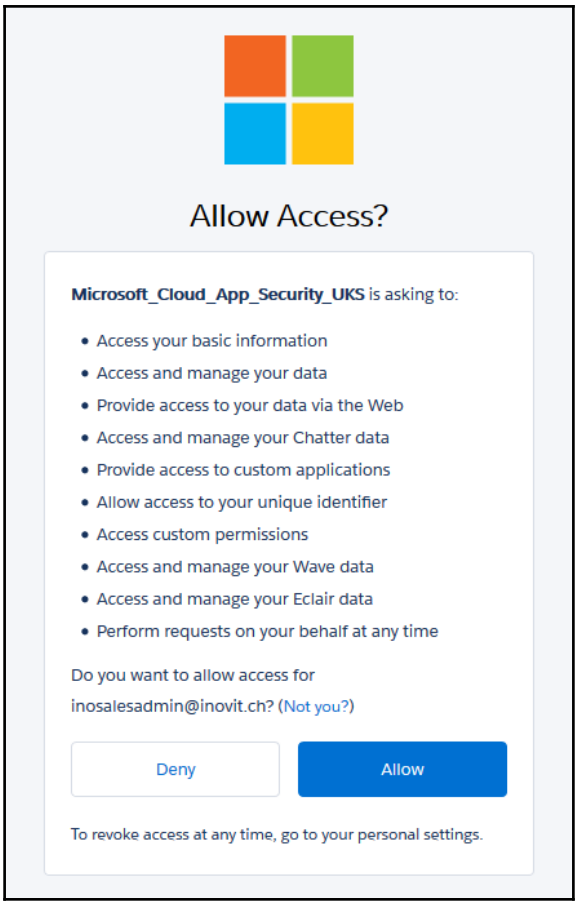


Roles in the OAuth framework

The provided list of roles are as follows:

- **Authorization Server**: Issues the access token
- **Resource Server**: Verifies and accepts the access token
- **Client**: The application that requests the access token
- **User (Resource Owner)**: Is the end user, who grants the permission to access the resource server with an access token

In the following screenshot, you will find an example consent, which is used with Cloud App Security that requires access to Salesforce:



Typical OAuth 2.0 consent

You will find more information about the Azure AD consent framework at: `https://docs.microsoft.com/en-us/azure/active-directory/develop/consent-framework`.

It is also important to understand the flows in OAuth 2.0 that define the process for obtaining an access token.

Main OAuth 2.0 flow facts

There are four flows defined in the specification:

- **Authorization code flow**:
    - One-time code issued to client
    - Client redeems code for access token
    - Access and ID token
    - Used for server-side apps
    - **Authorization code flow with proof key for code exchange** (**PKCE**) for native/mobile applications
- **Client credentials flow**:
    - Authenticates the client, not the user
    - Client receives an access token for itself
    - Does not support refresh tokens
    - Recommended for client applications with no end user (machine-to-machine communication)
- **Resource owner password flow**:
    - Client collects username/password from user
    - Exchange username/password for access token
    - Used if you control the client application and the resource
    - Typically used in online services, where the online service client app talks with the own service
- **Implicit flow**:
    - Client is untrusted (public)
    - No refresh token issued
    - Recommended to use for **Single Page Applications** (**SPA**)

You can use the following resources to deploy several OAuth 2.0 examples with your on-premise ADFS infrastructure:

```
https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/
development/enabling-oauth-confidential-clients-with-ad-fs
```

```
https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/
development/native-client-with-ad-fs
```

Work through the following examples with Azure AD to get deeper into the different flow types:

- **OAuth 2.0 implicit grant flow**: `https://docs.microsoft.com/en-us/azure/active-directory/develop/v1-oauth2-implicit-grant-flow`
- **OAuth 2.0 auth code grant flow**: `https://docs.microsoft.com/en-us/azure/active-directory/develop/v1-protocols-oauth-code`
- **OAuth 2.0 On-Behalf-Of flow**: `https://docs.microsoft.com/en-us/azure/active-directory/develop/v1-oauth2-on-behalf-of-flow`
- **OAuth 2.0 client credentials flow**: `https://docs.microsoft.com/en-us/azure/active-directory/develop/v1-oauth2-client-creds-grant-flow`

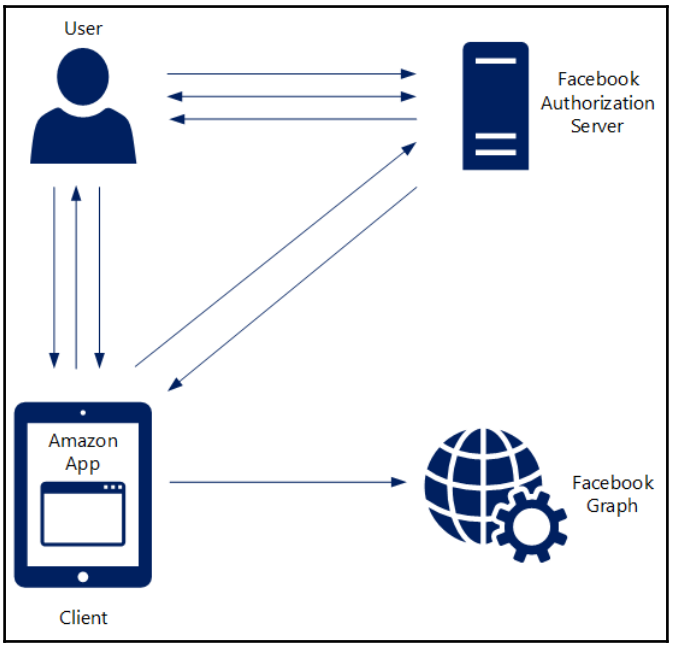
The next section provides you with information about the authorization code flow.

Authorization code flow

The main concept of this flow is that the client gets an authorization code first and uses the code to redeem the access token. It's recommended for private clients (web apps or native mobile applications) that have the capability to spawn a web browser. In that case, private clients establish a secret with the OAuth server. The secret will be used to authenticate the client during the access token redemption.

The access tokens expire and need to be refreshed with refresh tokens, and each of them has their own lifetime and can be stored for a longer term. A refresh token can also be used to redeem a new access token later.

The following figure shows an example of the authorization code grant type:



Authorization code grant type flow

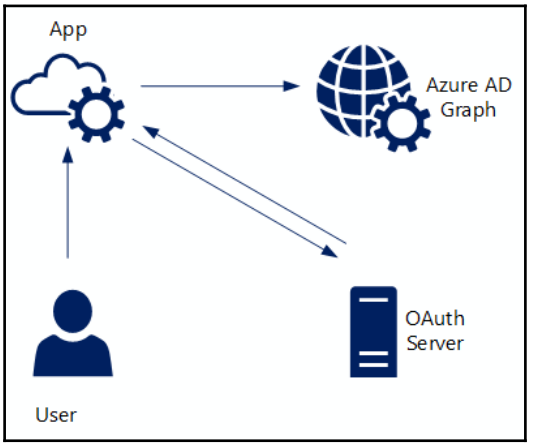
The flow works in the following order:

1. The **User** clicks the button to post this purchase
2. The **Client** redirects the user to OAuth server
3. The **User** authenticates and grants consent
4. The OAuth server redirects the **User** to the **Client** with the authorization code
5. The **Client** requests an access token from the **Authorization Server**
6. The OAuth server returns the access token to the **Client**
7. The **Client** uses the access token to authorize to the resource

Client credential flow

The main concept of this flow is application authentication and not user authorization, where the application establishes a secret. The application authenticates with the secret and receives an access token. Users are not involved in this flow and the client can perform this flow out of band.

The following diagram shows the client credential grant type:



Client credential grant flow

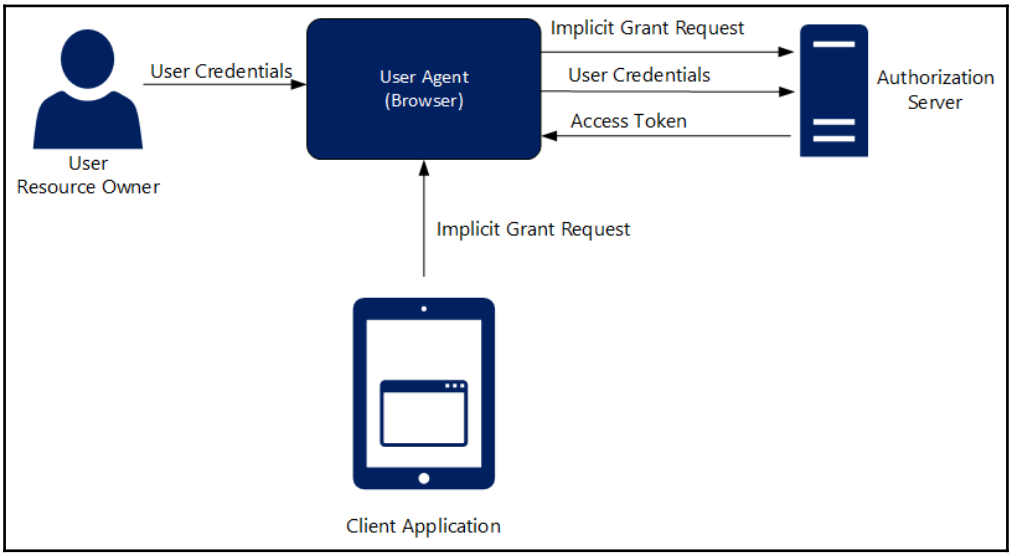
The flow runs in the following order:

1. The **User** uses the client (through the address book app)
2. The client authenticates to the OAuth server (**Azure AD**)
3. The OAuth server provides an access token to the client
4. The client calls resource (web service) with the access token in the header

Implicit grant flow

While acquiring an access token, this flow is mostly used by JavaScript clients running in the web browser. Also, important to note is that there is no authentication requirement for the JavaScript client. The difference between this and the authorization code flow is that the access token will be received in the grant request.

The following figure shows an example of the implicit grant type.



Implicit grant flow

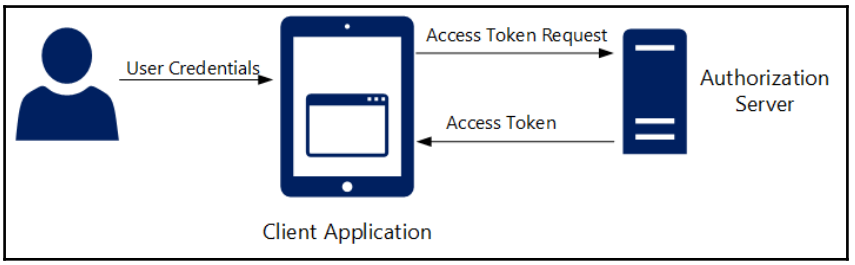
The flow in the main points runs as follows:

1. The **Client Application** opens a browser to send the user to the **Authorization Server**
2. The authorization prompt appears to the user and approves the application request
3. A redirect for the user goes back to the application with an **Access Token**

Resource owner password credentials flow

The main concept in this flow is that the resource owner must trust the client application. This means that the resource owner has to give its credentials directly to the client application.

The following figure shows the resource owner password credentials grant type:



Resource owner password credential flow

In conclusion, we can say that the specification of OAuth 2.0, which was published in October 2012, is now commonly used in rich client and modern application scenarios and with **RESTful Web API** access. The token format is agnostic but **JWT** is primarily used.

OpenID Connect (OIDC)

OIDC was established as a standard by its membership in February 2014.
OIDC provides a lightweight framework for identity interactions in a RESTful manner. The specification was developed under the OpenID Foundation and has its roots in OpenID; it was greatly affected by OAuth 2.0, because that specification was not intended for authentication. Microsoft was also a co-author of the OIDC specification.

Key facts about OIDC

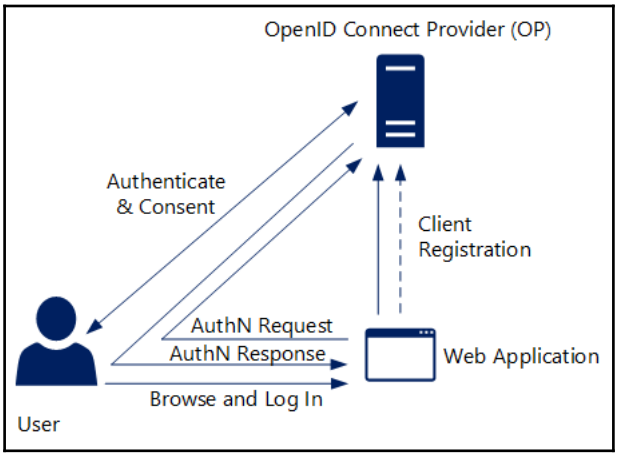
It defines the following identity layers on top of OAuth 2.0:

- It uses two OAuth 2.0 flows:
    - Authorization code flow
    - Implicit flow
- Adds an ID token to OAuth 2.0 exchange
- Adds the ability to request claims using an OAuth 2.0 access token

The following roles are used:

- **OpenID Connect Provider** (**OP**): Authorization server issues the ID token
- **Relying Party**: Client application that requests the ID token
- **ID token**: Issued by the OP
- **Claim**: Information about the user

The following figure shows the OpenID Connect flow:



OpenID Connect flow

The flow runs with the following steps:

1. A client registers with the OP
2. The user browses to the web app and initiates login
3. The web app redirects the user to the OP
4. The user authenticates to the OP and gives consent for the web app to use his identity
5. OP builds the authorization code
6. OP redirects the user back to the web app with the authorization code
7. The web app sends the authorization code to OP
8. OP creates the ID token and access token and sends back to the web app
9. The web app verifies the ID token

The specification also uses a UserInfo EndPoint with the following characteristics:

- Returns additional claims about a user
- REST-based endpoint
- Authenticates with access token received from OPx
- Response returned in JSON

More information about OIDC and Azure AD is available at: `https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-protocols-oidc`.

Build the following sample applications with an on-premise ADFS infrastucture:

- **Enabling OIDC**: `https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/development/enabling-openid-connect-with-ad-fs`
- **Single logout for OIDC**: `https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/development/ad-fs-logout-openid-connect`

Use the examples with Azure AD to dive further into the OIDC implementation. These are covered here: `https://docs.microsoft.com/en-us/azure/active-directory/develop/sample-v2-code`

In conclusion, we can say that the specification was released in February 2014, was co-authored by Microsoft, and is used for web sign-in when consent is needed. The token format is JWT.

Pass-through authentication and seamless SSO

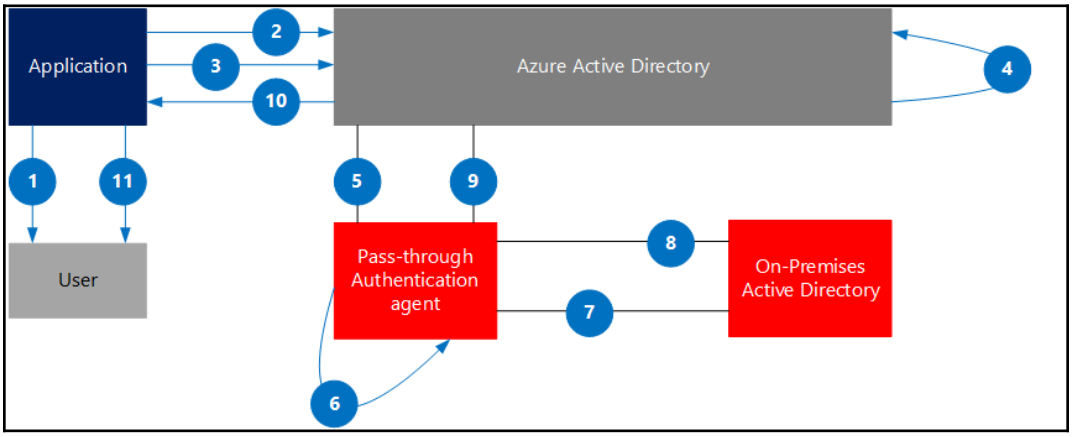
Azure AD pass-through authentication provides an alternative to the Azure AD password hash synchronization and a local ADFS infrastructure if all claims-based applications are connected to the Azure AD. Microsoft offers with this service the capabilities to reduce the on-premise complexity and operations of ADFS. Furthermore, in combination with the password hash synchronization, customers get a redundant and flexible authentication environment. You are also able to include password protection features for your local Active Directory.

Pass-through authentication supports the Azure AD conditional access policies, Azure MFA, and the blocking of legacy authentications to secure your organization's or customer environment. The communication of the on-premise agent and the Azure AD service is protected with certificate authentication. The feature can support multi forest infrastructures if forest trusts are enabled and the UPN-suffix routing is configured correctly. In combination with seamless SSO, users get a native SSO experience and are automatically signed into on-premise and cloud-based applications.

The following components are involved in the user sign-in process:

- **Azure AD STS**: **Stateless security token service** (**STS**) for processing sign-in requests and security token issuance
- **Azure Service Bus**: Communication component between cloud and on-premise
- **Azure AD Connect Authentication Agent**: Listener and responder to password validation requests
- **Azure SQL Database**: Storage for tenant associated authentication agents
- **Active Directory**: Store for local user accounts and passwords

Let's look at the following diagram to understand the functionality accessing the Outlook web app:



Path-through authentication flow

Deep dive information about the service can be found at the following source: `https://docs.microsoft.com/en-us/azure/active-directory/hybrid/how-to-connect-pta-security-deep-dive`

The flow runs with the following steps:

1. User tries to access Outlook web app
2. When the user is not signed in, he will be redirected to the Azure AD sign-in page
3. User enters his username, and selects **Next**
4. User enters his password, and selects **Sign In**
5. The Azure AD receives the sign in request and puts the username/password encrypted with the public key of the authentication agents in a queue
6. The on-premise authentication agent retrieves the encrypted credentials from the queue

The agent retrieves requests over a pre-established and persistent connection.

7. The agent decrypts the password with his private key
8. The agent validates the username and credential against the local Active Directory, like ADFS
9. The Domain Controller evaluates the request and responds with the result the agent
10. The agent responds back to Azure AD
11. Azure AD validates the answer—the user will be signed in or Azure MFA will be executed
12. If all works fine, the user is signed in

To choose the best option for your hybrid authentication solution, you can use the following source to help you with your decision `https://docs.microsoft.com/en-us/azure/security/azure-ad-choose-authn`.

Multi-factor authentication

Protecting sensitive information or application access with additional authentication is an important task, not just in the on-premise world. In particular, it needs to be extended to every sensitive cloud service used. There are a lot of variations for providing this level of security and additional authentication, such as certificates, smart cards, or biometric options. For example, smart cards depend on special hardware used to read the smart card and cannot be used in every scenario without limiting the access to a special device or hardware. The following table gives you an overview of different attacks and how they can be mitigated with a well-designed and implemented security solution:

Attacker	Possible security solution
Password brute force	Strong password policies
Shoulder surfing Key or screen logging	One-time password solution
Phishing or pharming	Server authentication (HTTPS)
Man-in-the-Middle Whaling (Social engineering)	Two-factor authentication Certificate or one-time password solution
Certificate authority corruption Cross Channel Attacks (CSRF)	Transaction signature and verification Non repudiation
Man-in-the-Browser Key loggers	Secure PIN entry Secure messaging Browser (read only) Push button (token) Three-factor authentication

Microsoft provides the Azure MFA functionality to address exactly the attacks described in the previous table. With a one-time password solution, you can build a very capable security solution to access information or applications from devices that cannot use smart cards as the additional authentication method. Otherwise, for small or medium business organizations, a smart card deployment, including the appropriate management solution, will be too costly, and the Azure MFA solution can be a good alternative for reaching the expected higher security level.

Azure MFA

Azure MFA provides the security of a two-step verification. The user gets many options to verify his identity, like phone calls, text messages, the mobile app with notifications, and created codes. Also, the integration of OATH hardware tokens is in the actual preview. To roll out the feature, you can use the following sources:

```
https://docs.microsoft.com/en-us/azure/active-directory/authentication/
tutorial-mfa-applications
```

```
https://docs.microsoft.com/en-us/azure/active-directory/authentication/
howto-mfa-getstarted
```

Azure MFA includes the capabilities to integrate on-premise components with the following two agents:

- **Active Directory Federation Services** (**ADFS**) 2016 and higher
- **Network Policy Server** (**NPS**)

If you need to integrate older versions of ADFS or third-party RADIUS servers, the only option is to deploy the on-premise Azure MFA server. With this integration there are some limitations, such as:

- No synchronization of the registration information
- Usage of two registration and user portals
- Additional MFA infrastructure to manage

In the next section, we will discuss the certificate authentication option.

Certificate authentication

Azure AD supports the use of certificate-based authentication. Devices can use a client certificate to connect. This feature eliminates the need to enter a username/password combination to access your mail and Microsoft applications on your mobile device. You can use the following two sources to gather more information on the specific platform:

- **Android**: `https://docs.microsoft.com/en-us/azure/active-directory/authentication/active-directory-certificate-based-authentication-android`
- **iOS**: `https://docs.microsoft.com/en-us/azure/active-directory/authentication/active-directory-certificate-based-authentication-ios`

Otherwise, you can use the device authentication as an alternative method. We describe the device authentication in the next section.

Device authentication

The other method that Microsoft includes in the multi-factor solution framework is the device authentication. The device authentication can be used with the following registration or join method of the device:

- Azure AD registration
- Workplace Join
- Azure AD Join
- Hybrid Azure AD Join

You will find more information about all the different options at the following source: `https://docs.microsoft.com/en-us/azure/active-directory/devices/overview`. You can use these states for conditional access in your on-premise ADFS or your Azure AD.

To deploy on-premise conditional access with your ADFS infrastructure, you can use the following source: `https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/operations/configure-device-based-conditional-access-on-premises`. For cloud environments, we recommend the following source to start your journey into conditional access: `https://docs.microsoft.com/en-us/azure/active-directory/conditional-access/overview`.

Biometric authentication

Windows Hello is another authentication solution that uses biometric user information like the fingerprint, face, or iris recognition. The functionality is available for personal or enterprise-grade security to provide authentication. You also can use this feature against your on-premise ADFS and your Azure AD. Use the following source to plan and deploy **Windows Hello** in your organization: `https://docs.microsoft.com/en-us/windows/security/identity-protection/hello-for-business/hello-planning-guide`.

Summary

Working through the chapter and the given references, you have worked through many aspects and practical examples of the most important authentication protocols. We have tried to provide you with a very crisp reference card with much valid additional information about this topic. You should be able to use WS-Federation, SAML, OAuth 2.0, and OIDC in your design or configuration work for customers or your own organization. As already mentioned at the beginning of the chapter, we will use the knowledge in the upcoming chapters.

In the next `Chapter 7`, *Deploying Solutions on Azure AD and ADFS*, we start to work on it directly!

7
Deploying Solutions on Azure AD and ADFS

What's better than using theory directly in a practical lab activity? Nothing, in our eyes. Working through `Chapter 6`, *Managing Authentication Protocols,* you learned about the different authentication methods used in current environments. Now, we'll start to use this knowledge to deploy several scenarios to our **Azure AD** and **Active Directory Federation Services** (**ADFS**). We will help you to understand all the configuration steps for required a suitable authentication environment.

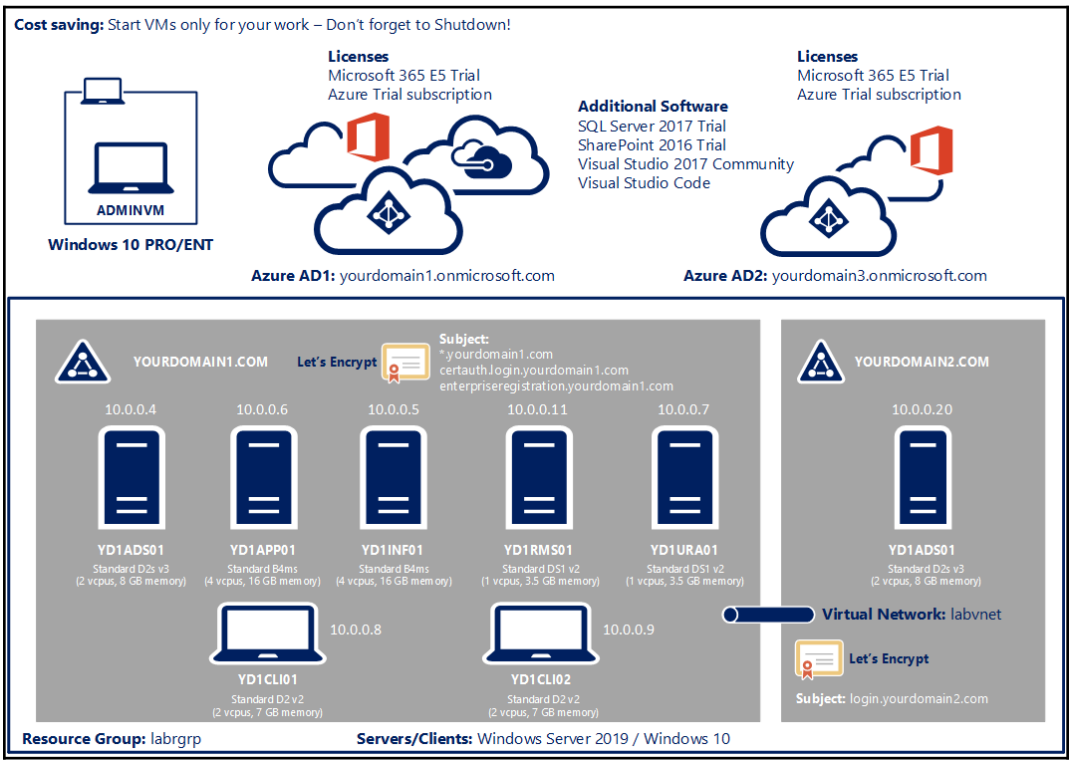
In this chapter, you will extend your current lab environment, install and configure the service we connect, and configure your authentication solution to handle different methods. We use this approach so that you can understand all the stuff from scratch. We highly recommend that you read `Chapter 6`, *Managing Authentication Protocols*, before you run through this chapter, which will cover the following topics:

- Basic environment installation and configuration
- Azure AD authentication deployments
- ADFS authentication deployments
- Integrating Azure MFA into ADFS

Are you ready to go? Yes! We'll start by preparing our environment.

Basic environment installation and configuration

In this chapter, we start to extend our actual simulated on-premises infrastructure with the additional servers we need in order to demonstrate and configure different capabilities. In the following diagram, we introduce the complete environment we'll have configured after working through all the labs in the book:



Lab environment overview

In this chapter, we will add `YD1APP01` and `YD1URA01` to our environment. `YourDomain1` (`YD1`) is used to identify the machine in the correct domain. In our case, we used `INODEMOAPP01` as an example. You need to provision the machines with the previous values.

You already deployed the `YDADS01` domain controller in `Chapter 2`, *Understanding Identity Synchronization*. For all future virtual servers, use the same **Azure subscription**, the same **resource group**, and the same **virtual network**. Join the virtual machines to your existing Active Directory. For the domain controller installation, we used `inovitlabs.ch` as an example. In the following chapters, we will use similar DNS suffixes. To be clear, we use `inovitdemos.ch` as a continuous representation of `inovitlabs.ch` and we will add `azureid.ch` and `leano.ch` for the different scenarios.

`azureid.ch` will be used for the second domain we use without any cloud integration.
`leano.ch` will be used as a cloud-only environment for business to business communication.
You can use the names you prefer.

For a better understanding, we show one example for an additional virtual machine, and you should run into the following result if you have provisioned all virtual machines by the end of the book:

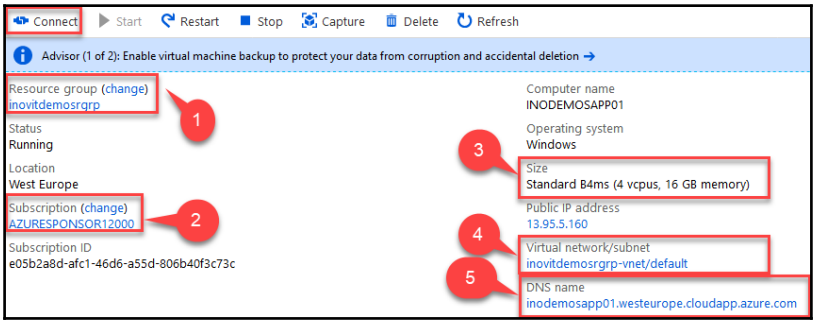
INOAZUREIDADS01	Running	inovitdemosrgrp	West Europe	AZURESPONSOR12000
INODEMOSADS01	Running	inovitdemosrgrp	West Europe	AZURESPONSOR12000
INODEMOSAPP01	Running	inovitdemosrgrp	West Europe	AZURESPONSOR12000
INODEMOSCLI01	Running	inovitdemosrgrp	West Europe	AZURESPONSOR12000
INODEMOSCLI02	Running	inovitdemosrgrp	West Europe	AZURESPONSOR12000
INODEMOSINF01	Running	inovitdemosrgrp	West Europe	AZURESPONSOR12000
INODEMOSRMS01	Running	inovitdemosrgrp	West Europe	AZURESPONSOR12000
INODEMOSURA01	Running	inovitdemosrgrp	West Europe	AZURESPONSOR12000

Virtual machine overview

For **INODEMOSAPP01** (`YDAPP01`), we use a Standard B4ms (4 VCPUs, 16 GB memory) virtual machine with Windows Server 2019. This virtual machine will be the host for running an SQL Server instance, most of our demo apps, and Visual Studio. For this reason, we use such a big machine type:

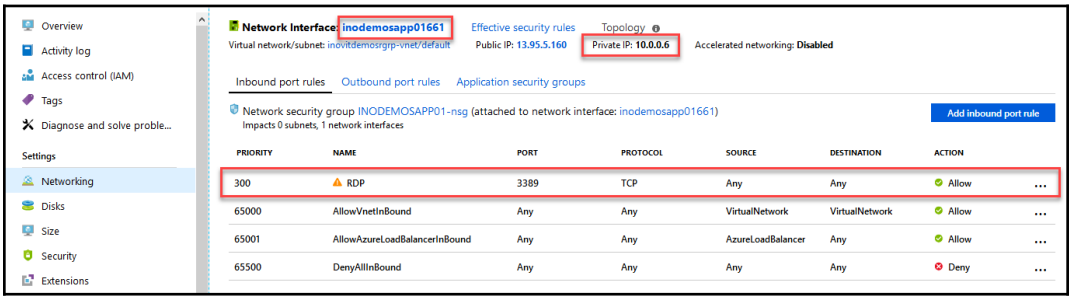
- Always use the same resource group
- Always use the same subscription
- Use the size from the previous screenshot (recommended)
- Always use the same virtual network
- Always configure a DNS name for a virtual machine

You can use the **Connect** button to download the RDP-connection file:



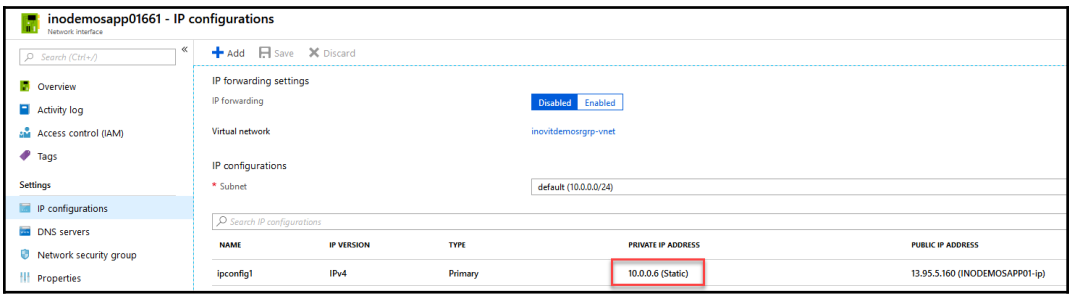
VM configuration options

To make the labs as easy as we can, we define for every machine an **RDP** inbound rule at creation time, and configure a static internal IP address for the virtual machine:



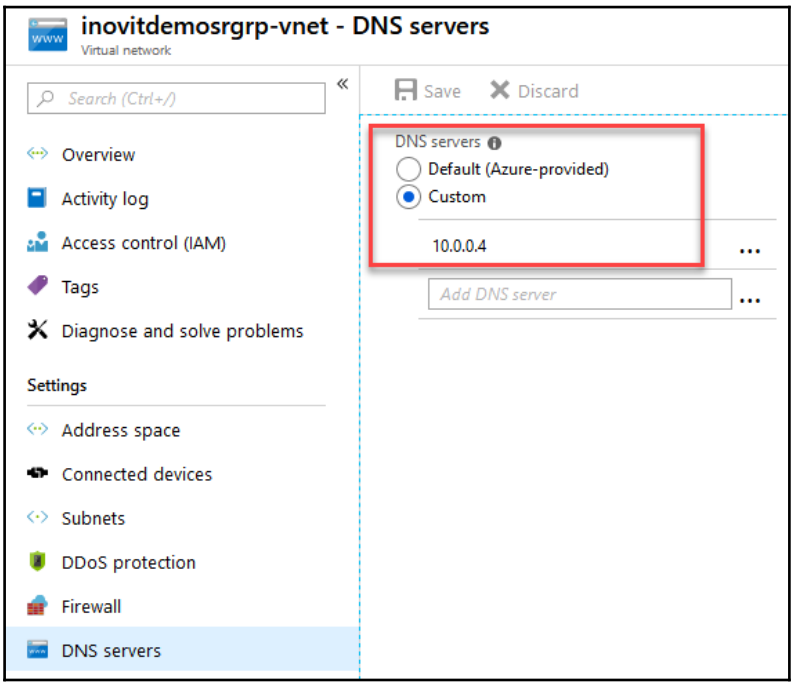
Firewall configuration

If you click on **Network Interface**, you can configure the static IP:



Static IP configuration

Next, we need to click on the virtual network to configure our `YDADS01` domain controller as the primary **DNS server**:

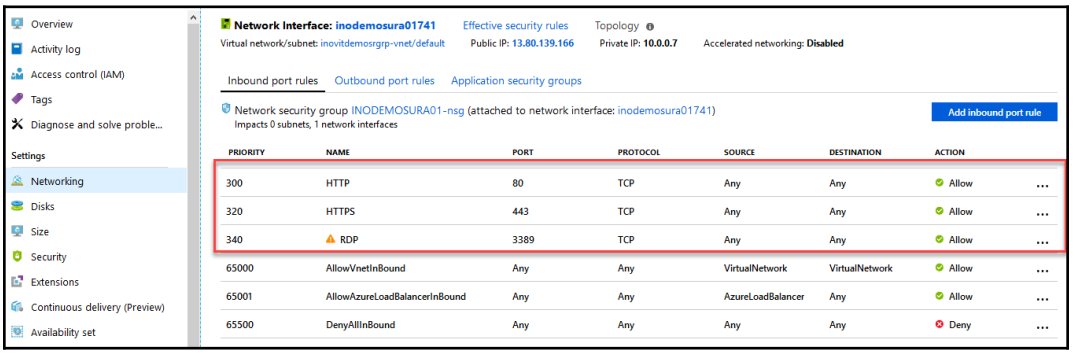


Custom DNS configuration

Now we are ready to go, and you can follow the same steps for any additional virtual machines. We also provide a scripting solution in the code package of the book.

This configuration is just for lab and demonstration purposes, and not for a production environment. We highly recommend that you do the same on your already provisioned domain controller configuration.

For this chapter, and specifically the `YD1URA01` virtual machine, we need to configure two additional inbound port rules to provide external access to our `Web Application Proxy`. We need to open port **80 TCP** (**HTTP**) and **443 TCP** (**HTTPS**). The expected configuration should look like the following:



Firewall configuration

Now that we have discussed the virtual machine setup, you should deploy the `YD1APP01` and `YD1URA01` servers to run the following labs in this chapter.

Create the certificate for your environment with let's encrypt

To provide a valid certificate for our tasks, we need to install the `Posh-ACME` PowerShell module to use let's encrypt as our certificate provider. The certificates are valid for three months, without any cost. If you want to use your environment for longer, you just need to renew the certificate.

Find the module description and further information at the following source: `https://docs.microsoft.com/en-us/office365/enterprise/base-configuration-dev-test-environment`

1. With the following command, you install the module:

```
# Install for all users (requires an elevated PowerShell)
Install-Module -Name Posh-ACME

# Install for current user
Install-Module -Name Posh-ACME -Scope CurrentUser
```

2. After installing the PowerShell module, we can request our certificate (replace the DNS suffix with your value):

```
New-PACertificate
'*.inovitdemos.ch','certauth.login.inovitdemos.ch' -
AcceptTOS -Contact jochen.nickel@inovit.ch
```

3. You will get a message that you need to create the `TXT` entries in your public DNS server required for the domain verification process:

```
Please create the following TXT records:
----------------------------------------
_acme-challenge.inovitdemos.ch -> OtORHpZ1CruSu2Tb-iZ1oSnU7oMOsiT0fMPq8pDDeVM
_acme-challenge.certauth.login.inovitdemos.ch -> 5je--CO7tiMYFXdclcFQKu9UwxnbQTVuzV4U8xpIBQs
----------------------------------------
```

Let's encrypt verification entries

4. Create the values in your public DNS like in the following example:

```
_acme-challenge.inovitdemos.ch   TXT   15MIN TTL
OtORHpZ1CruSu2Tb-iZ1oSnU7oMOsiT0fMPq8pDDeVM
_acme-challenge.certauth.login.inovitdemos.ch  TXT  15MIN
TTL  5je--CO7tiMYFXdclcFQKu9UwxnbQTVuzV4U8xpIBQs
```

Keep in mind that it requires some time for the values to become active. After creating the entries, you can press any key to continue.

5. After a successful validation, you'll get a message that you can remove the values from your public DNS:

```
Please remove the following TXT records:
----------------------------------------
_acme-challenge.inovitdemos.ch -> OtORHpZ1CruSu2Tb-iZ1oSnU7oMOsiT0fMPq8pDDeVM
_acme-challenge.certauth.login.inovitdemos.ch -> 5je--CO7tiMYFXdclcFQKu9UwxnbQTVuzV4U8xpIBQs
----------------------------------------
```

Removal note for Let's encrypt verification option

6. Your new certificate is created:

```
Subject            NotAfter              KeyLength Thumbprint                              AllSANs
-------            --------              --------- ----------                              -------
CN=*.inovitdemos.ch 4/12/2019 3:45:09 PM 2048      B7440A0C8D7BADB3E740BE01F18A461E411FAB3D {*.inovitdemos.ch, certauth.login.inovitdemos.ch}
```

Newly created certificate information

7. Now, we need to find and import the certificate on every server in our environment. Use the following cmdlet:

```
Get-PACertificate | fl
```

The output of the preceding command is as follows:

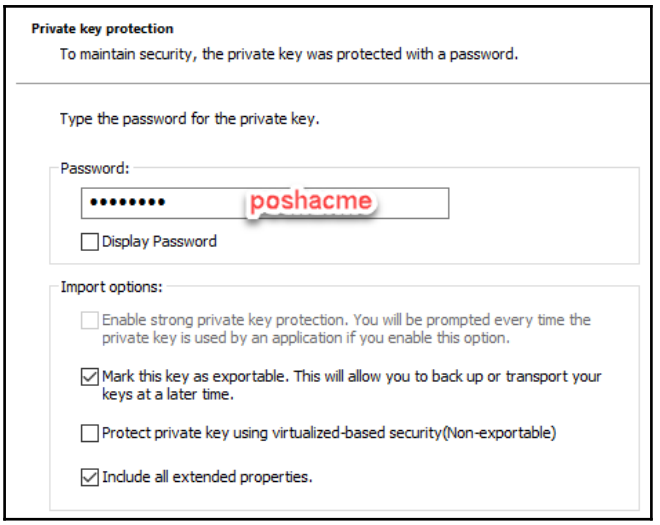
```
Subject       : CN=*.inovitdemos.ch
NotBefore     : 1/12/2019 2:45:09 PM
NotAfter      : 4/12/2019 3:45:09 PM
KeyLength     : 2048
Thumbprint    : B7440A0C8D7BADB3E740BE01F18A461E411FAB3D
AllSANs       : {*.inovitdemos.ch, certauth.login.inovitdemos.ch}
CertFile      : C:\Users\jochen.nickel\AppData\Local\Posh-ACME\acme-v02.api.letsencrypt.org\49385785\!.inovitdemos.ch\cert.cer
KeyFile       : C:\Users\jochen.nickel\AppData\Local\Posh-ACME\acme-v02.api.letsencrypt.org\49385785\!.inovitdemos.ch\cert.key
ChainFile     : C:\Users\jochen.nickel\AppData\Local\Posh-ACME\acme-v02.api.letsencrypt.org\49385785\!.inovitdemos.ch\chain.cer
FullChainFile : C:\Users\jochen.nickel\AppData\Local\Posh-ACME\acme-v02.api.letsencrypt.org\49385785\!.inovitdemos.ch\fullchain.cer
PfxFile       : C:\Users\jochen.nickel\AppData\Local\Posh-ACME\acme-v02.api.letsencrypt.org\49385785\!.inovitdemos.ch\cert.pfx
PfxFullChain  : C:\Users\jochen.nickel\AppData\Local\Posh-ACME\acme-v02.api.letsencrypt.org\49385785\!.inovitdemos.ch\fullchain.pfx
PfxPass       : System.Security.SecureString
```

Certificate information

8. Just type `.\cert.pfx` and hit *Enter,* and then follow the import wizard.
9. Choose **Local Machine** for the storage location:

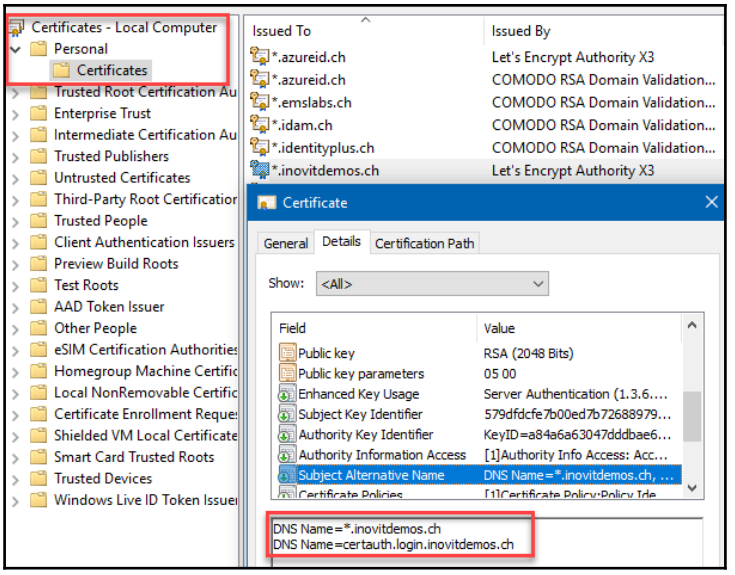
Certificate Import Wizard

10. The default password used from the PowerShell module to protect the `PFX` file is `poshacme`. Choose **Mark key as exportable** only if you want to export the file with your own password to use it on other servers:



Certificate import options

11. Now, open the certificate management console for the local machine store with the `certlm.msc` command and you are able to verify the installed **Certificate**:



Certificate store and Subject Alternative Names

We don't show this procedure in every following chapter, so be aware that you always install the certificate on a new server in your environment.

Now that we have finished all the preparation tasks, we can install `ADFS` and the `Web Application Proxy` for our demo application installations.

Installing the ADFS farm on YDADS01

In our environment, we will install the `ADFS` farm on our domain controller. In production, this is not always the recommended option, but for cost saving, we'll use this option. Let's start!

1. Create the internal DNS entry for the `ADFS` farm:

```
Add-DnsServerResourceRecord -ZoneName "inovitdemos.ch" -A
-Name "login" -IPv4Address "10.0.0.4"
```

2. Create the internal DNS entries for the demo applications:

```
Add-DnsServerResourceRecord -ZoneName "inovitdemos.ch" -A
-Name "claims" -IPv4Address "10.0.0.6"
Add-DnsServerResourceRecord -ZoneName "inovitdemos.ch" -A
-Name "kerb" -IPv4Address "10.0.0.6"
Add-DnsServerResourceRecord -ZoneName "inovitdemos.ch" -A
-Name "basic" -IPv4Address "10.0.0.6"
Add-DnsServerResourceRecord -ZoneName "inovitdemos.ch" -A
-Name "ntlm" -IPv4Address "10.0.0.6"
```

3. Create the `ADFS` **group managed service account** (**gMSA**):

```
Add-KdsRootKey -EffectiveTime (Get-Date).AddHours(-10)
New-ADServiceAccount svcadfs -DNSHostName
login.inovitdemos.ch -ServicePrincipalNames
http/login.inovitdemos.ch
```

4. Install the `ADFS` farm:

```
# Get the certificate thumbprint
gci Cert:\LocalMachine\My\ | fl
# Install the ADFS Farm with WID
Install-WindowsFeature ADFS-Federation -
IncludeManagementTools
Install-AdfsFarm -CertificateThumbprint
```

```
66F1BF8CCD904DF74154A5D24769DE155E874257 -
FederationServiceName login.inovitdemos.ch -
GroupServiceAccountIdentifier inovitdemos\svcadfs$ -
FederationServiceDisplayName "INOVITDEMOS Login"
# Server restart
Restart-Computer
```

5. Enable `IdP Initiated Signon Page`:

```
Set-AdfsProperties -EnableIdPInitiatedSignonPage $true
```

6. Check your `ADFS` infrastructure installation, opening the provided links in your browser:

```
# Check ADFS Metadata
https://login.inovitdemos.ch/adfs/fs/federationserverservi
ce.asmx
# Check IDP Sign In
# Add "login.inovitdemos.ch" to the "Local Intranet"
settings in IE
https://login.inovitdemos.ch/adfs/ls/IdpInitiatedSignon.as
px
```

7. Now, we connect `ADFS` to our Azure AD and change to a federated environment. We use the `SupportMultipleDomain` options to be ready for later demonstrations, where we integrate a second domain suffix and an additional Active Directory forest:

```
Connect-MsolService
Convert-MsolDomainToFederated -DomainName inovitdemos.ch -
SupportMultipleDomain:$true
```

8. Next, we install and configure the `Web Application Proxy`.

Installing the Web Application Proxy on YD1URA01

We will install the `Web Application Proxy` on `YD1URA01` to provide external access to our `ADFS` infrastructure by following these steps:

1. Open an evaluated PowerShell instance and use the following commands:

```
# Get the certificate thumbprint
gci Cert:\LocalMachine\My\ | fl
```

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

# Configure Web Application Proxy
$creds = Get-Credential
Install-WebApplicationProxy -FederationServiceName
"login.inovitdemos.ch" -FederationServiceTrustCredential
$creds -CertificateThumbprint
"66F1BF8CCD904DF74154A5D24769DE155E874257"

# Server restart
Restart-Computer
```

2. Next, we will publish Active Directory `Federation Services` with the `Web Application Proxy`:

```
Add-WebApplicationProxyApplication -Name "Federation
Services" -BackendServerUrl
"https://login.inovitdemos.ch/" -ExternalUrl
"https://login.inovitdemos.ch/" -ExternalPreauthentication
"PassThrough" -ExternalCertificateThumbprint
"66F1BF8CCD904DF74154A5D24769DE155E874257"
```

3. Now, we need to create the entries in the public DNS for `ADFS` and the demo applications, where you need to use your DNS server names:

```
login.inovitdemos.ch. CNAME 15 MIN
inodemosura01.westeurope.cloudapp.azure.com
basic.inovitdemos.ch. CNAME 15 MIN
inodemosura01.westeurope.cloudapp.azure.com
kerb.inovitdemos.ch. CNAME 15 MIN
inodemosura01.westeurope.cloudapp.azure.com
ntlm.inovitdemos.ch. CNAME 15 MIN
inodemosura01.westeurope.cloudapp.azure.com
claims.inovitdemos.ch. CNAME 15 MIN
inodemosura01.westeurope.cloudapp.azure.com
```

4. Next, we will test the external `ADFS` functionality.
5. Use your administrative virtual machine or any other external client, open `https://login.inovitdemos.ch/adfs/ls/idpinitiatedsignon.aspx` in a browser, and log in with one of your test users.

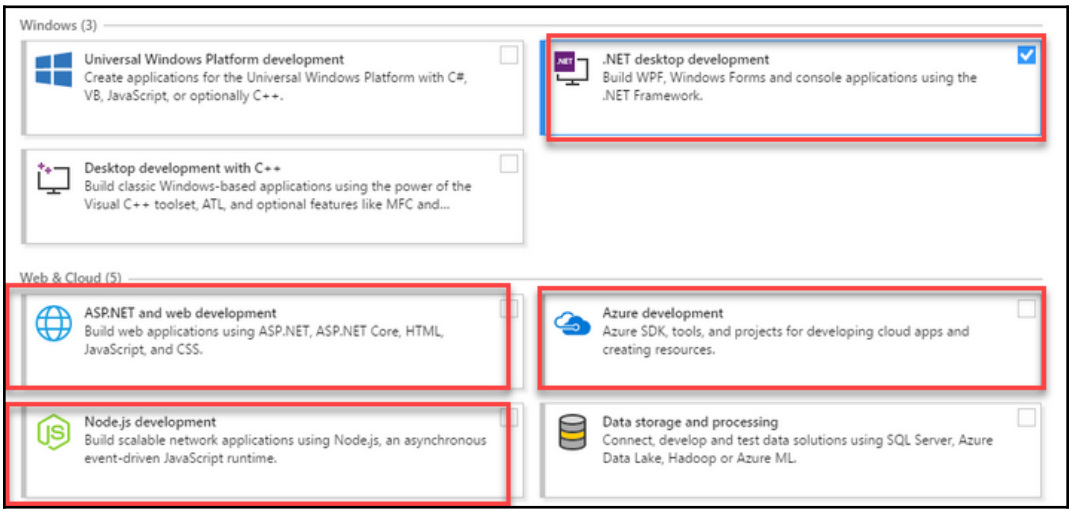
Now, we have a working `ADFS` and `Web Application Proxy` infrastructure. The only missing actors are the demo applications that we will implement now on the `YD1APP01` server and subscribe to several cloud services.

Installing demo applications on (YD1APP01) for ADFS

For our tasks in this and later chapters, we need to install a SQL Server and Management Tools instance on the server.

The following software needs to also be installed on the server:

- Visual Studio (`https://bit.ly/2NzB14g`) with the following options:



Visual Studio package installation options

- Visual Studio Code (`https://code.visualstudio.com/`)

Next, we start with the **Internet Information Service** (**IIS**) base installation:

1. Install IIS:

```
Install-WindowsFeature NET-Framework-Core,NET-
Framework-45-Features,Web-Mgmt-Console,Web-Asp-Net,Web-
Asp-Net45,Web-Basic-Auth,Web-Client-Auth,Web-Digest-
```

```
Auth,Web-Dir-Browsing,Web-Dyn-Compression,Web-Http-
Errors,Web-Http-Logging,Web-Http-Redirect,Web-Http-
Tracing,Web-ISAPI-Ext,Web-ISAPI-Filter,Web-Lgcy-Mgmt-
Console,Web-Metabase,Web-Mgmt-Console,Web-Mgmt-
Service,Web-Net-Ext,Web-Net-Ext45,Web-Request-Monitor,Web-
Server,Web-Stat-Compression,Web-Static-Content,Web-
Windows-Auth,Web-WMI,Windows-Identity-Foundation
```

2. Now, we install the SQL instance. Create the SQL organizational unit in your Active Directory:

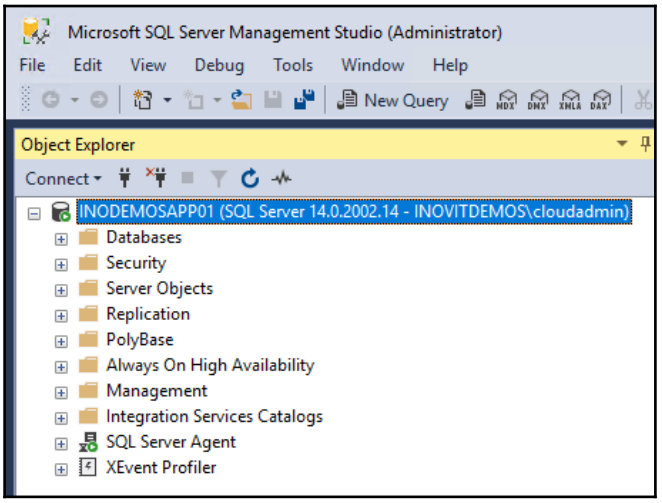
```
New-ADOrganizationalUnit -Name "SQL" -Path "OU=Managed
Service Objects,DC=INOVITDEMOS,DC=CH"
New-ADOrganizationalUnit -Name "Users" -Path
"OU=SQL,OU=Managed Service Objects,DC=INOVITDEMOS,DC=CH"
```

3. Create the service accounts:

```
# SQL Database engine service account
New-ADUser -Name "svcsqldb" -SamAccountName svcsqldb -
UserPrincipalName svcsqldb@inovitdemos.ch -path
"OU=Users,OU=SQL,OU=Managed Service
Objects,DC=inovitdemos,DC=ch" -AccountPassword (ConvertTo-
SecureString "YourPassword" -AsPlainText -Force) -
Description "SQL Database Engine" -Enabled $True

# SQL Agent service account
New-ADUser -Name "svcsqldbagent" -SamAccountName
svcsqldbagent -UserPrincipalName
svcsqldbagent@inovitdemos.ch -path
"OU=Users,OU=SQL,OU=Managed Service
Objects,DC=inovitdemos,DC=ch" -AccountPassword (ConvertTo-
SecureString "YourPassword" -AsPlainText -Force) -
Description "SQL Database Engine Agent" -Enabled $True
```

4. Next, we can do a standard installation of SQL Server 2017 (Trial Edition: `https://bit.ly/2FoSE6z`) with a default instance. You can follow the guidance at `https://bit.ly/2lKJ9mi` to install the service.
5. Just use the following parameters to install:
    - In general, use the default settings
    - Add just the database engine and the Full-Text Index as features
    - Add the current user as administrator
    - Use the service accounts for the database engine and the SQL Agent; choose the start type to be automatic
    - Download and install **SQL Management Studio**

6. After the installation, you should be able to connect to the new SQL instance with **SQL Management Studio**:



Connecting to the SQL instance with SQL Management Studio

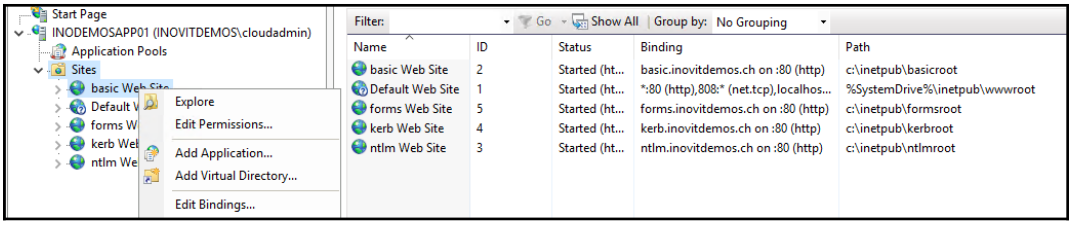
We use the test site from the Microsoft blog at `https://bit.ly/2QDIqQN` (credit: Emmanuel Boersma), to demonstrate the different authentication methods.

7. Configure the authentication demo apps.
   You need to change the bold values to create all demo apps with the following values:
   - `basic`
   - `kerberos`
   - `ntlm`

8. Use the following script:

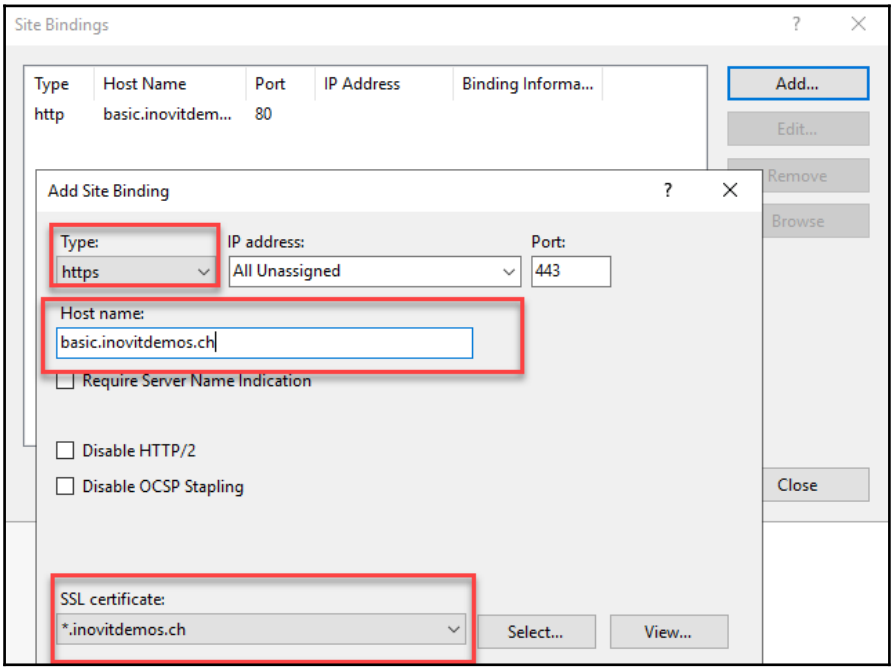
```
New-Item C:\inetpub\basicroot -type Directory
Import-Module Webadministration
cd IIS:
New-Item 'IIS:\Sites\Basic Web Site' -bindings
@{protocol="http";bindingInformation=":80:basic.inovitdemo
s.ch"} -physicalPath 'c:\inetpub\basicroot'
```

9. Next, will add an HTTPS binding to every demo app:



IIS demo app configuration overview

10. Do this for every app!



IIS binding configuration

Now, we need to work on some extra tasks for the applications.

We start with the claims app:

1. Use the claims demo app package from the code package and execute the script (change it to your values before execution):

```
.\deploy-testsite.ps1 -SourcePath
"C:\inovit\deploy" -SiteName "claims Web Site" -
SitePhysicalPath C:\inetpub\claimsroot -ADFSServer
inodemosads01 -AppFQDN claims.inovitdemos.ch
```

2. Connect the HTTPS binding of the claims demo app to your certificate.

Let's do the tasks for the `kerberos` app:

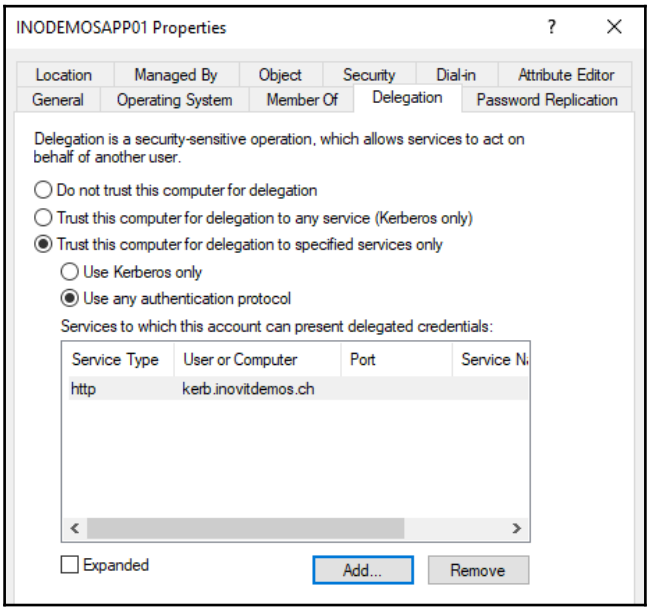
1. Create and configure a service account on`YD1ADS01` to run the application pool with the name `svckrbapp`, and with the password never expires option selected. Note the password so you can provide it in the following configurations:

```
New-ADUser -Name "svckrbapp" -SamAccountName svckrbapp -
UserPrincipalName svckrbapp@inovitdemos.ch -path
"OU=Users,OU=AAD,OU=Managed Service
Objects,DC=inovitdemos,DC=ch" -AccountPassword (ConvertTo-
SecureString "YourPassword" -AsPlainText -Force) -
Description "Kerberos App Pool Account" -Enabled $True
```

2. Next, we need to configure the correct Service Principal Name for the Kerberos Delegation scenario later:

```
setspn -S http/kerb.inovitdemos.ch inovitdemos\svckrbapp
```

3. Configure the Kerberos delegation on your `YD1APP01` and `YD1URA01` computer account (for external access) with the `svckrbapp` user account:



Kerberos Delegation configuration in AD

4. Next, we need to jump to your `YD1APP01` server to configure the Kerberos application.
5. Under **Application pools**, click **Add a new application pool**.
6. Name the pool Kerberos website and leave the other default values.
7. Click **Advanced Settings** and change the identity to the `svckrbapp` account.
8. Click on the Kerberos website and click **Advanced Settings**.
9. Change the application pool to the newly created one: Kerberos website.
10. Under **IIS Authentication**, enable **Windows Authentication** and disable **Anonymous Authentication**.
11. Go to **Windows Authentication** | **Advanced Settings** | Clear **Enable Kernel-mode**.
12. Copy the content of the `authpage` folder from the code package to the Kerberos root directory at `C:\inetpub\kerbroot`.

Next, we will configure the `ntlm` app:

1. Click on the NTLM website and click **Advanced Settings**
2. Change the application pool to Kerberos website
3. Under **IIS Authentication**, enable **Windows Authentication** and disable **Anonymous Authentication**
4. Go to **Windows Authentication** | **Advanced Settings** | **Clear Enable Kernel-mode**
5. Go to **Windows Authentication** | **Advanced Settings** | **Providers and leave only NTLM**
6. Copy the content of the `authpage` folder from the code package to the NTLM root directory at `C:\inetpub\ntlmroot`

Next, we will configure the `basic` app:

1. Copy the content of the `authpage` folder from the code package to the basic root directory at `C:\inetpub\basicroot`
2. In the IIS manager, go to the `basic` app and configure the authentication so that just the basic authentication is enabled

Reboot your `YD1ADS01`, `YD1APP01`, and `YD1URA01` servers before we move on to the next steps.

Now that we have finished the on-premises demo apps, we need to sign up for some cloud demo apps.

Subscribing to demo apps (Azure AD)

To complete all the authentication and information protection labs, we need to register for the following cloud applications:

- Sign up for a *salesforce developer account: `https://developer.salesforce.com/signup`
- Sign up for a service now developer account: `https://bit.ly/2FniUOC`
- Sign up for a business dropbox account trial: `https://bit.ly/1KYht6o`
- Sign up for a LinkedIn account: `https://bit.ly/1k7JbKB`
- Sign up for a *Twitter account: `https://bit.ly/1k7JbKB`

- Sign up for a Google account: **`https://bit.ly/2reCBP3`**
- Sign up for a Okta developer account: `https://bit.ly/2IOw3ix`
- Sign up for a Proxyclick trial: `https://bit.ly/2D5BKqV`

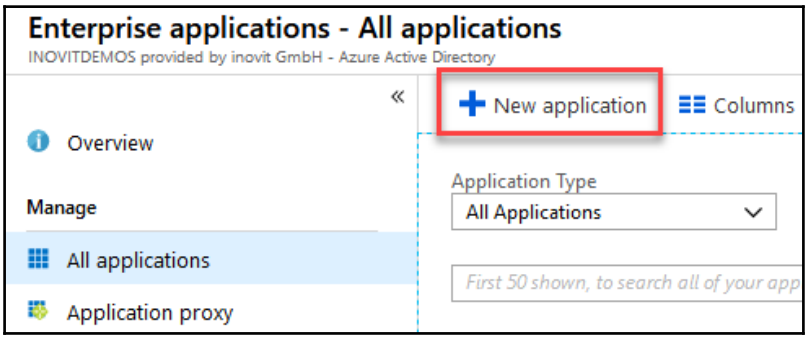
The applications marked with * will be used in this chapter and the other one in the `Chapter 8`, *Using Azure AD App Proxy and Web Application Proxy* and the following.

After the successful trial or developer accounts, we can start with our journey through the capabilities of Azure AD and ADFS.

Azure AD authentication deployments

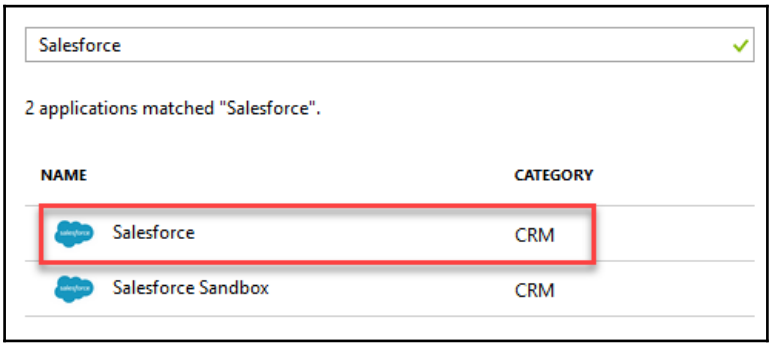
In this section, we will build applications for our users and work through the different authentication mechanisms provided by Azure AD. All the configurations we do in this section will be done with global administrator rights and on the Azure portal, `https://portal.azure.com`. We will start with Salesforce configuration:

1. Launch the Azure Active Directory blade and click **Enterprise applications**.
2. Under **All applications**, click **New application**:



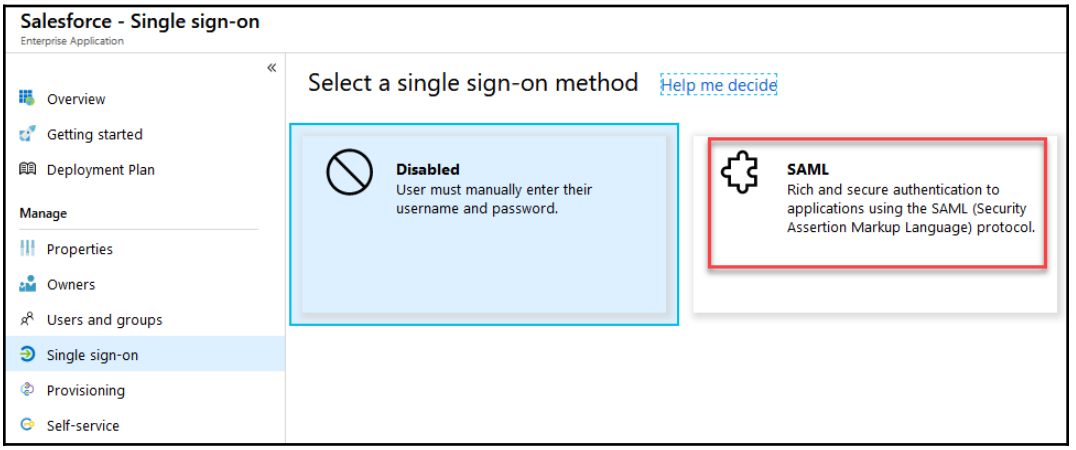
New application creation context

3. Type `Salesforce` in the search field:



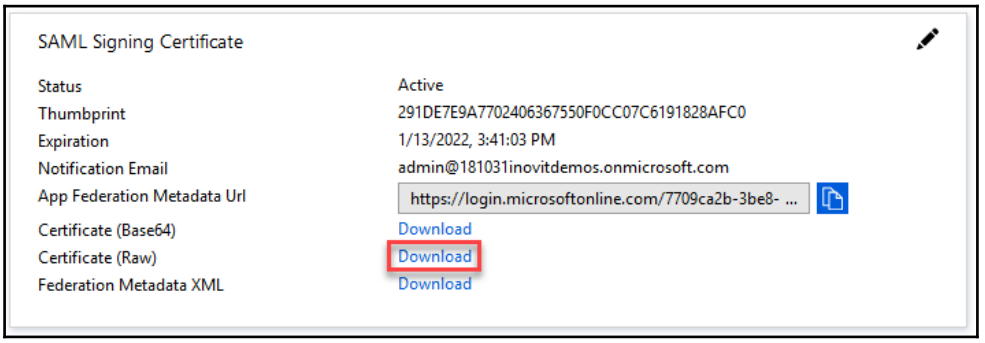
Salesforce enablement

4. Under **Single sing-on**, change to **SAML** authentication:



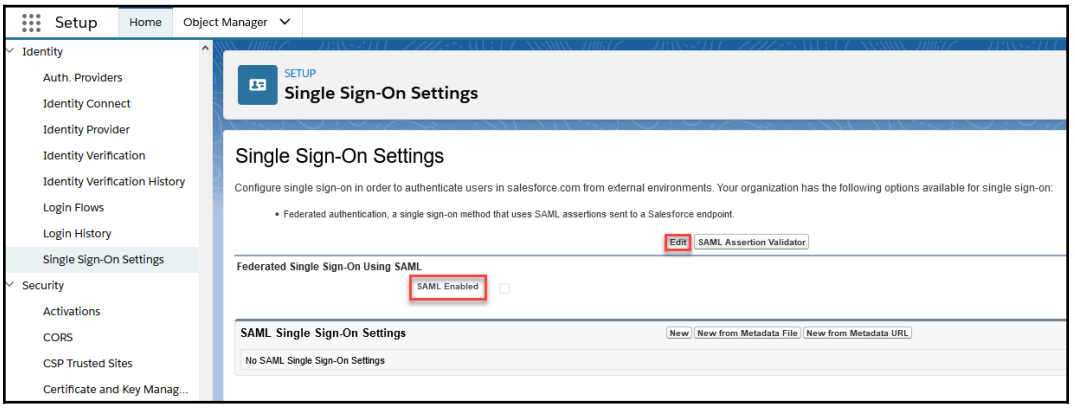
Choosing SAML as the authentication method

5. Go to the **SAML Signing Certificate** section and click **Download** on **Certificate (RAW)**:



Downloading the signing certificate

6. Now, log in to your Salesforce account and navigate to **Identity** | **Single Sign-On Settings**.
7. Edit the **SAML settings** and click **SAML Enabled**:

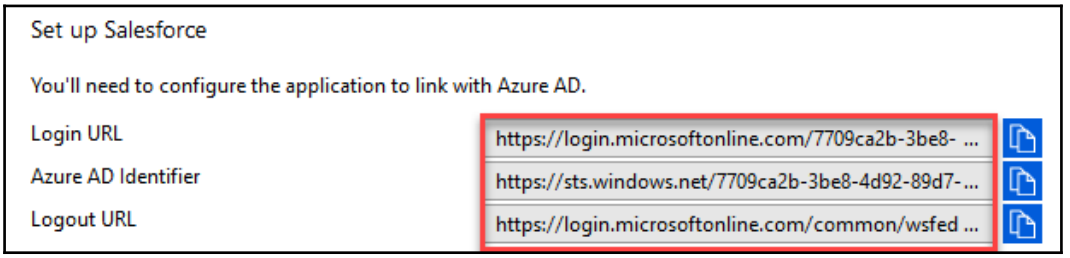


Configuration of SAML in Salesforce

8. Next, we will create new **SAML Single Sign-On Settings**; click **New**:

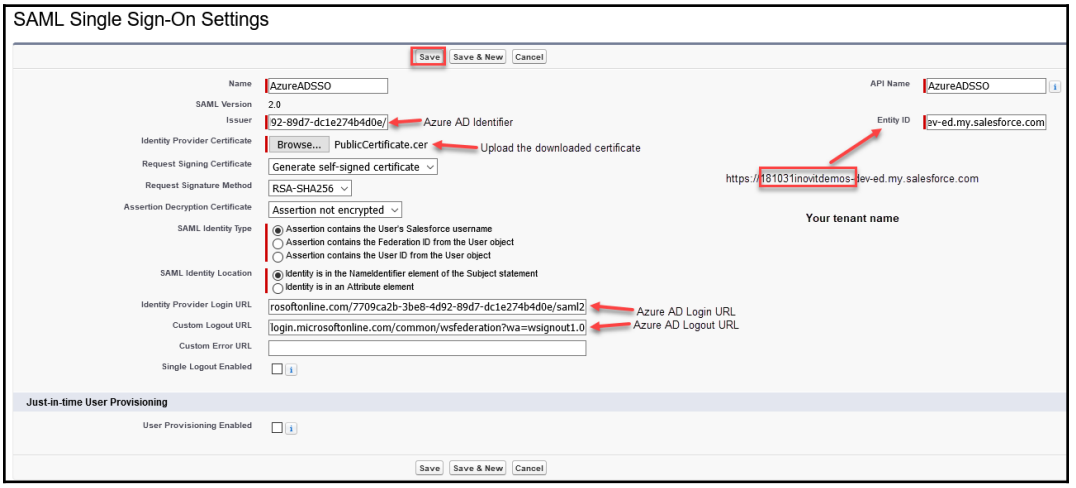
New settings dialog

9. To gather the values for the configuration, you need to jump back to your Azure portal and copy the three links to Notepad:



Salesforce configuration information about the Azure AD endpoints

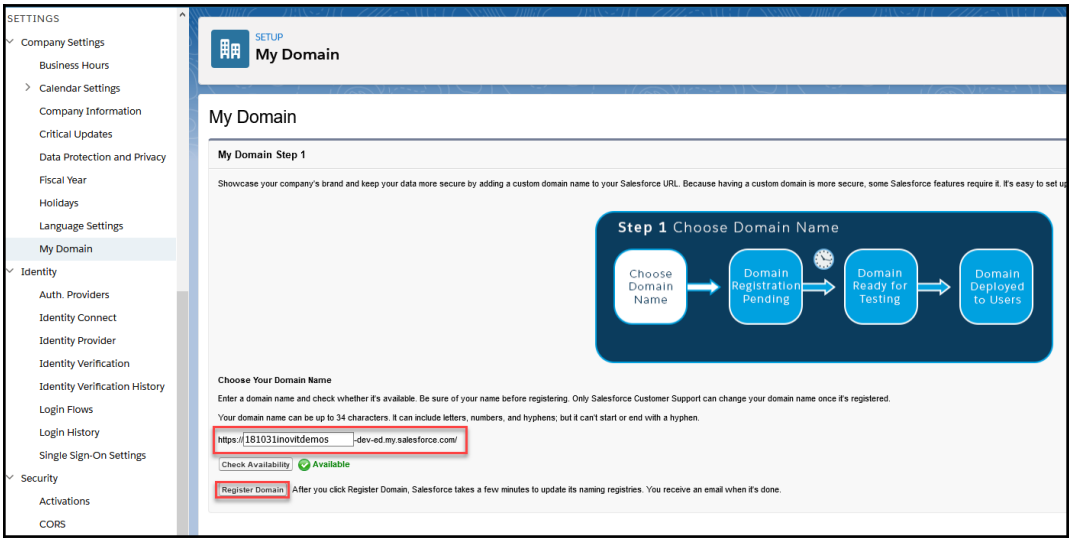
10. Fill in the following information on the Salesforce configuration page:



Salesforce SAML configuration page

11. Next, we need to configure our Salesforce domain name under **SETTINGS** | **Company Settings** | **My Domain**.

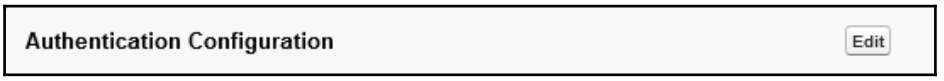
12. Use your tenant name and click **Check Availability** and **Register Domain**:



Salesforce domain registration process

The registration process takes about 5-10 minutes.

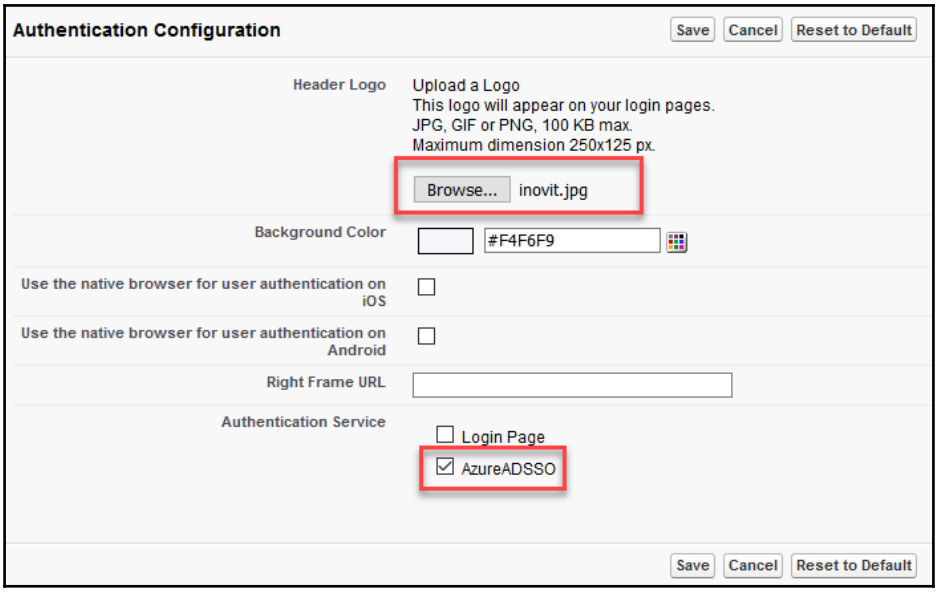
13. Refresh the page and **Edit** the **Authentication Configuration**:



Authentication configuration dialog

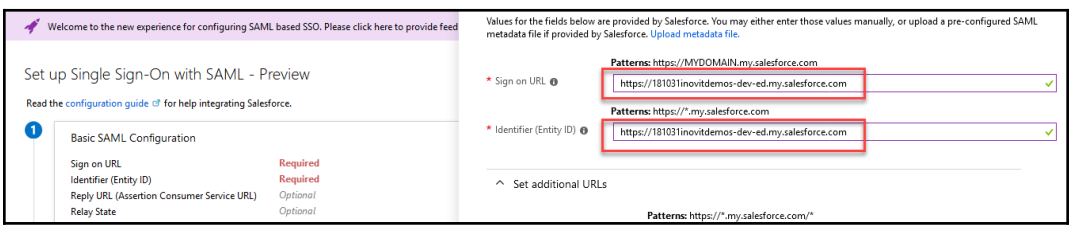
14. Click **Open** and **Upload a logo** from `Chapter 1`, *Building and Managing Azure Active Directory*.

15. Check **AzureADSSO** and **Save**:



Choosing the Authentication service AzureADSSO

16. Click on **login** and if you are prompted to register your phone, click **I Don't want ...**.
17. If you are prompted, log in with your Salesforce administrator account.
18. Next, under the **My Domain** section, click **Deploy to Users** and **OK**.
19. Now, we switch back to our **Azure AD** configuration.
20. Set the **Sign on URL** and **Identifier (Entity ID)** text box values to your value, `https://<TENANT>-dev-ed.my.salesforce.com`:



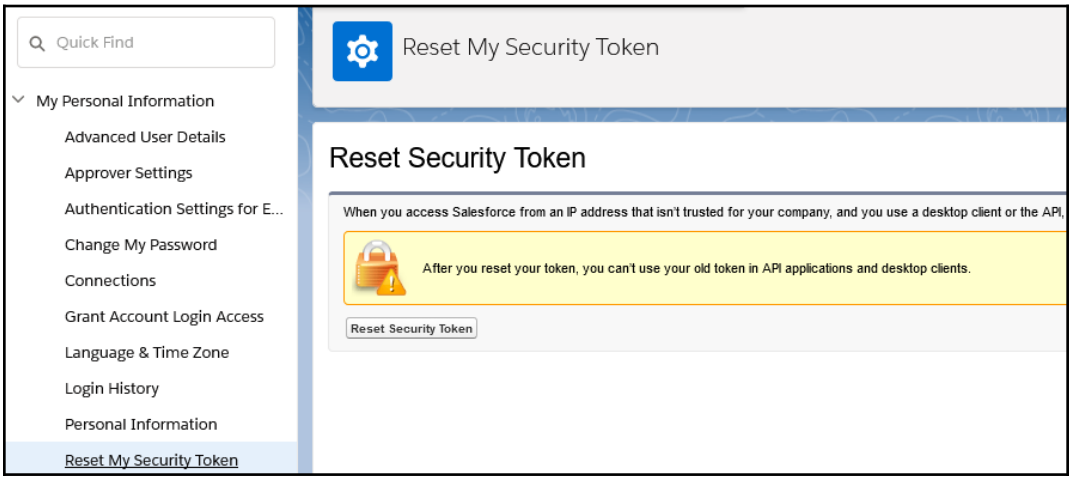
Azure AD Salesforce SAML configuration

Now that we have configured SAML authentication, we can activate user provisioning with Salesforce by following these steps:

1. Under the **Manage** section, click **Provisioning**.
2. **Provisioning Mode** drop-down list, set **Automatic.**
3. Under **Admin Credentials**, type in the admin username and password for accessing Salesforce.
4. Obtain a secret token by switching to the Salesforce administration:

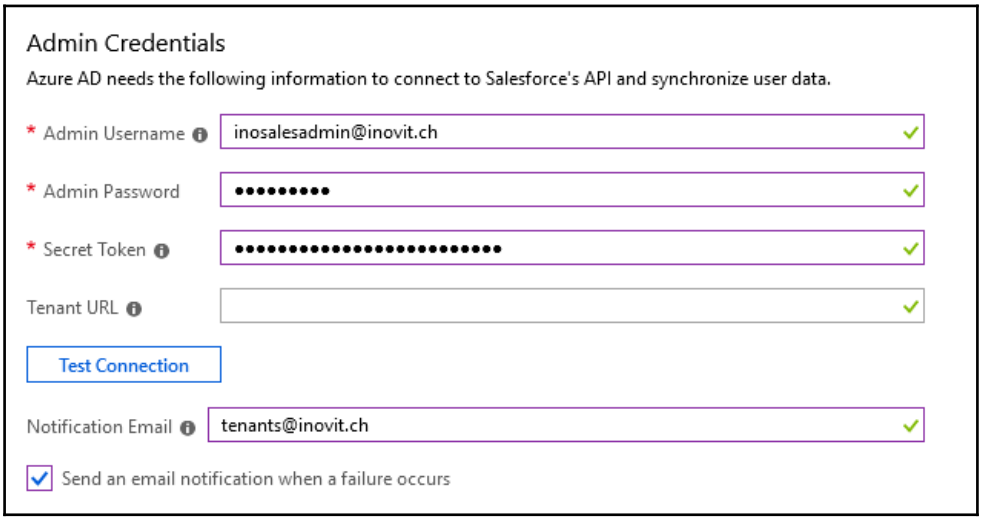
Secret token creation for provisioning

5. Click **Reset Security Token** and you will receive a new security token by mail:



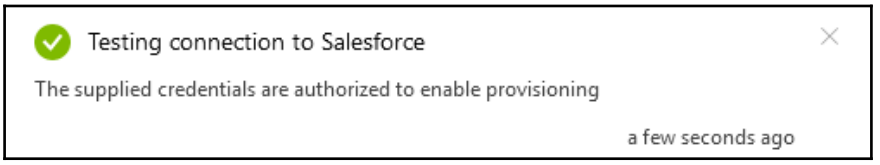
Get new secret token

6. Next, we configure the provisioning settings in the Azure portal. Use the token from your mailbox and configure a notification email address:



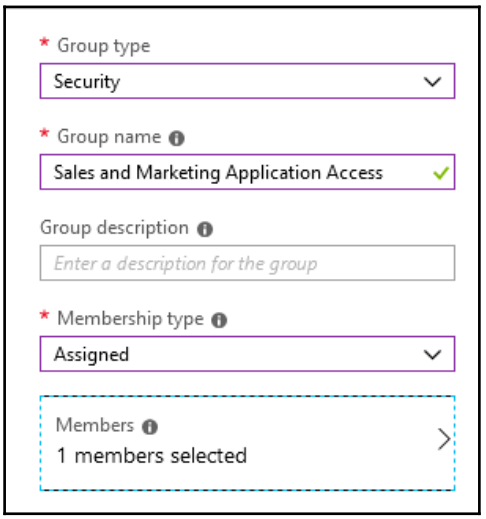
Configuring the provisioning service

The following message is expected:



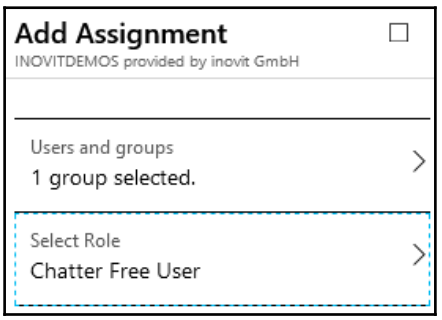
Test the connection to the Salesforce provisioning endpoint

7. To use the newly deployed application, we need to create and assign a group to the Salesforce application.
8. Create the following group and assign a licensed user from the Sales department:



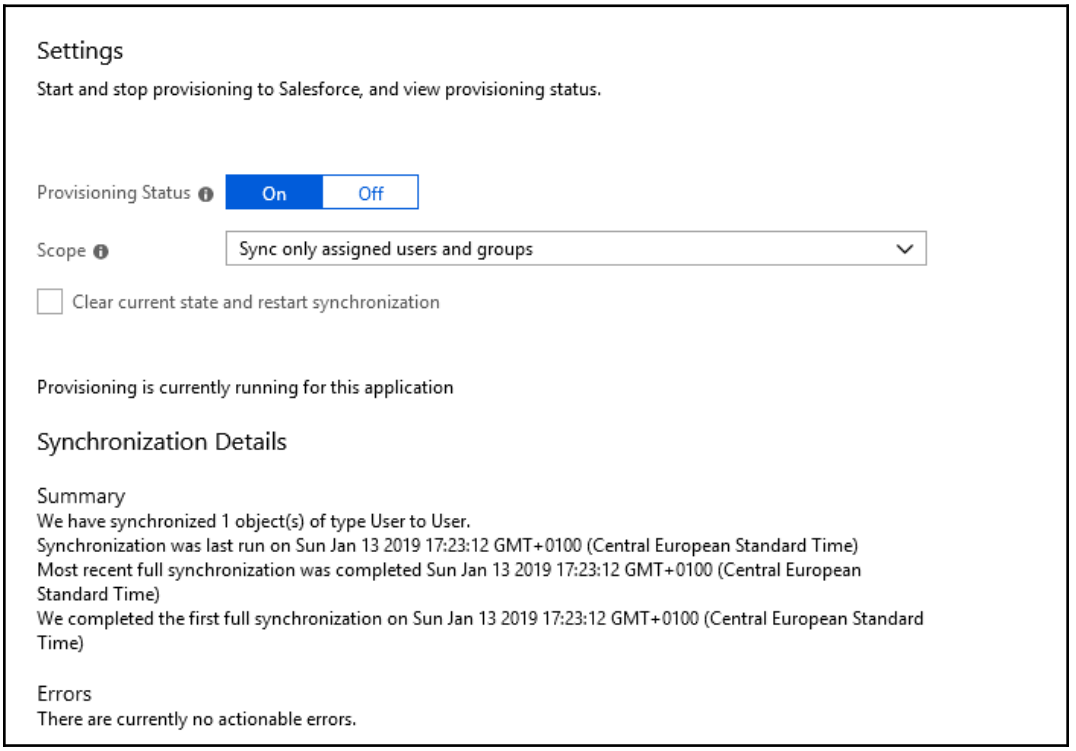
Group assignment for Salesforce app access

9. Assign this group with the following values:



Role selection

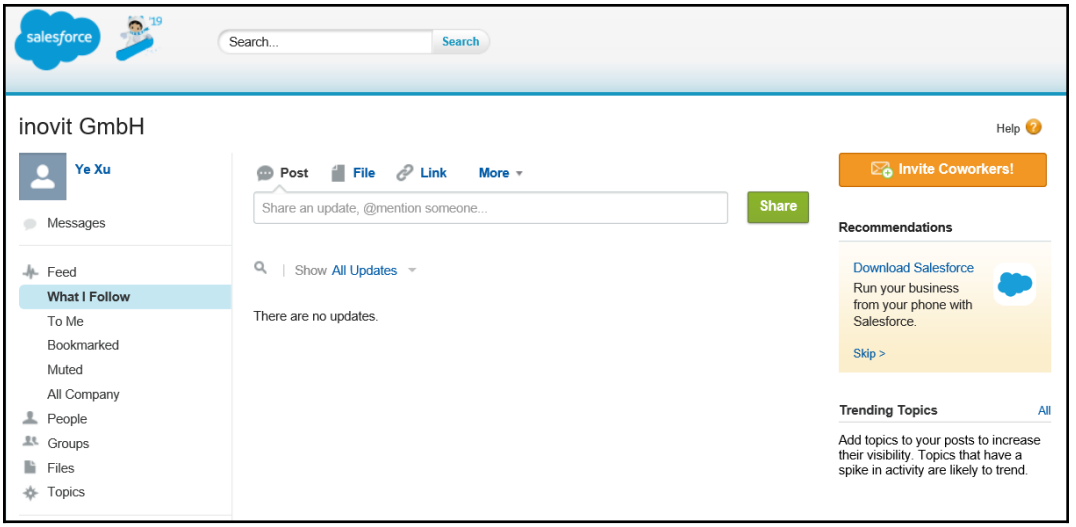
10. Under **Manage** | **Provisioning** | **Settings**, set **Provisioning Status** to **On** and leave the default **Scope**.
11. Click the checkbox for **Clear current state and restart synchronization** and click **Save**.
12. In the **Restart Synchronization** window, click **Yes**:



Synchronization and Provisioning status information

13. Test the application with your assigned test user at `https://myapps.microsoft.com`.

14. The following result is expected, a successful logon:

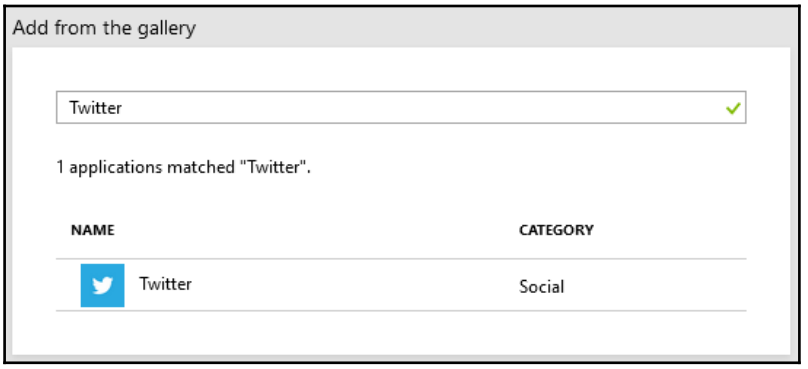


Successful logon on Salesforce with test user

We successfully deployed Salesforce, including SAML and provisioning capabilities, to our Azure AD.

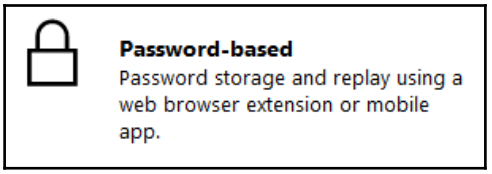
Now, we will use another feature in Azure AD with Twitter. For this, we use the password-based **Sign-In** option:

1. First, we need to add the **Twitter** app from the application gallery. You already know the process from Salesforce:



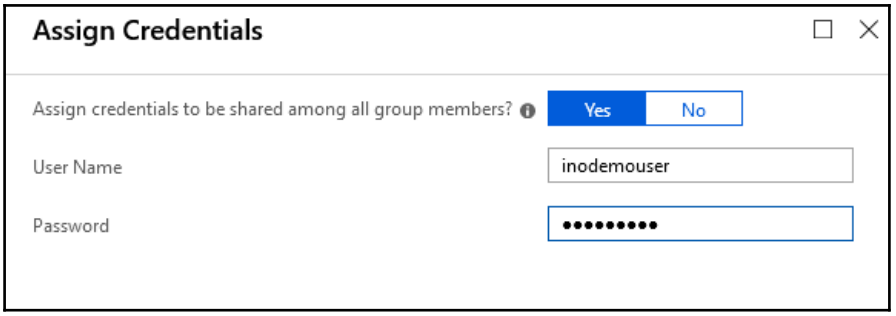
Adding Twitter to app catalog

2. Next, we choose **Password-based Single Sign-on** mode from the **Single Sign-On** section.
3. The wizard automatically sets the correct URL to Twitter:



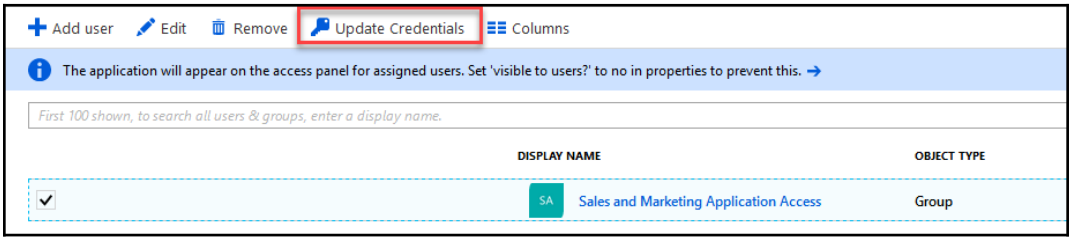
Choosing password-based authentication option

4. We assign the sales and marketing application access group to the application, and the provide the credentials we want to use and hide from the user:



Assigning the Twitter credentials to a group

5. You are also able to **Update Credentials** on the assigned groups:



Update credentials option

6. Now that we have configured the **Twitter** app for our sales and marketing users, you can test the functionality with the user over at `https://myapps.microsoft.com`.
7. You should have a **Single Sign-On** experience.

Some applications require access to the application's access panel (`https://myapps.microsoft.com`). In this case, the website requires a browser extension:

1. To configure Microsoft Edge for the access panel extension, launch your browser and navigate to `https://myapps.microsoft.com`
2. Log in as a test user
3. Click on **Twitter**
4. Click **Install Now**
5. Complete the installation wizard to install the **My Apps Secure Sign-In Extension**
6. You get a new extension notification; click **Turn it on**
7. Relaunch the browser and navigate to `https://myapps.microsoft.com`
8. Log in as a test user

This is our first impression of Azure AD's capabilities. We will dive deeper in the next `Chapter 8`, *Using Azure AD App Proxy and Web Application Proxy*, and now start to configure our first applications in our ADFS infrastructure.

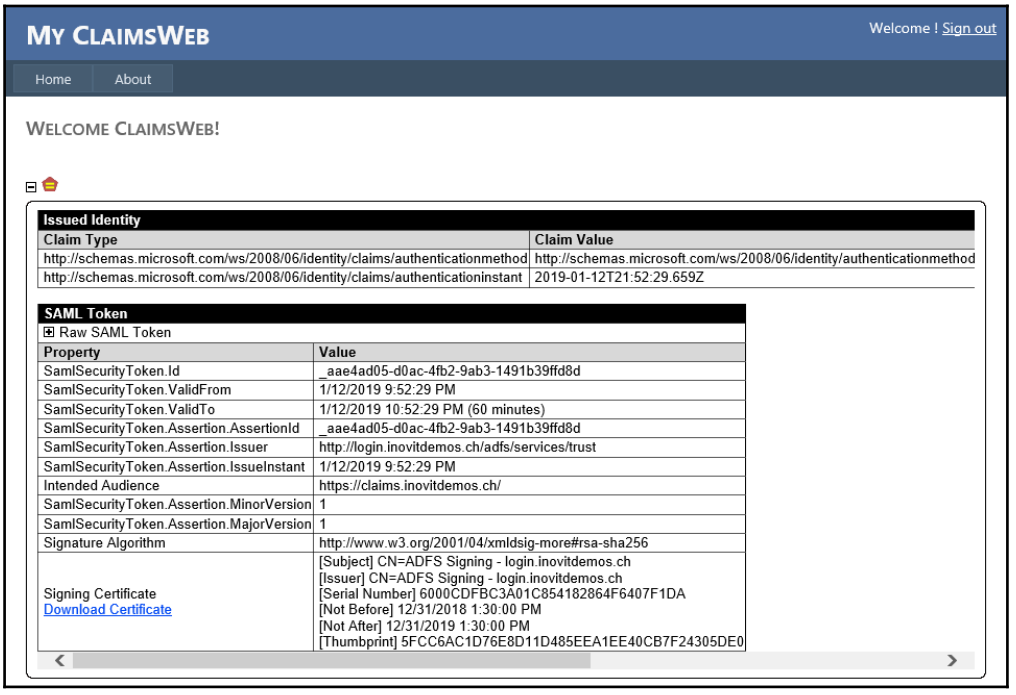
ADFS Authentication deployments

To configure a claims-based application with **WS-Federation**, we can use our claims demo application. With this application, you can test many features of ADFS with claims authentication and learn in a more practical way. Run the following configuration on your **YD1ADS01**. Later, we'll configure the application to get more experience:

1. Go to **Server Manager**, click **Tools**, and open ADFS Management.
2. Expand **Trust Relationships** and select **Relying Party Trusts**.
3. Select **Actions**, add **Relying Party Trust**, and click **Start**.
4. In the box, type `https://claims.inovitdemos.ch`:

ADFS relying party trust configuration

5. Click **Next**.
6. Type the display name as `claims Demo Web Site` and click **Next**.
7. Select **I do not want to configure multi-factor authentication settings for this relying party trust currently** and click **Next**.
8. Select **Permit all users to access this relying party**, and click **Next** | **Next**.
9. Clear the **Open the Edit Claim Rules** dialog box for this relying party trust when the wizard closes and click **Close**.
10. Verify the new app with Internet Explorer by typing `https://claims.inovitdemos.ch`:



Claims demo web site

The installation script automatically puts the thumbprint of the `Token-Signing` certificate in the `web.config` file of the claims web application. If you renew the `Token-Signing` certificate, you need to update the thumbprint in the application configuration.

You can gather the `Token-Signing` certificate from our ADFS instance with the following command:

```
Invoke-Command -ComputerName INODEMOSADS01 -ScriptBlock {Get-
ADFSCertificate}
```

The output of the preceding command will look as follows:

```
CertificateType : Token-Signing
IsPrimary       : True
StoreLocation   : CurrentUser
StoreName       : My
Thumbprint      : 5FCC6AC1D76E8D11D485EEA1EE40CB7F24305DE0
```

Get the Token-Signing certificate information

Another good helper for several test scenarios such as SAML, WS-Federation, OAuth2, and the strong authentication methods is the **Claims X-Ray tool** from Microsoft, which you can find at `https://bit.ly/2EqoPOi`, and it's quite easy to configure by following these steps:

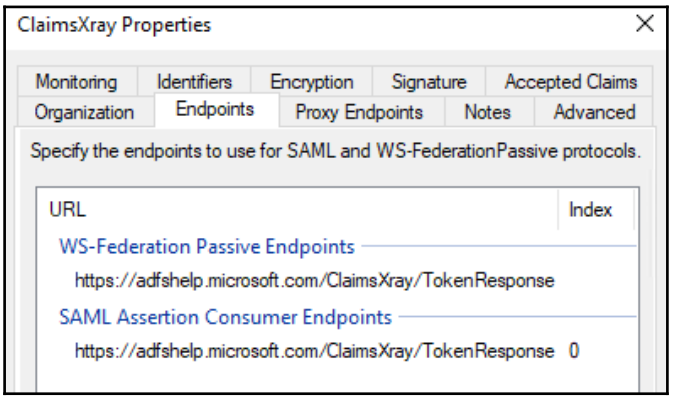
1. Open an evaluated PowerShell instance on your **YD1ADS01** and run the following script:

```
$authzRules = "=>issue(Type =
`"http://schemas.microsoft.com/authorization/claims/permit
`", Value = `"true`"); "
$issuanceRules = "@RuleName = `"Issue all
claims`"`nx:[]=>issue(claim = x); "
$redirectUrl =
"https://adfshelp.microsoft.com/ClaimsXray/TokenResponse"
$samlEndpoint = New-AdfsSamlEndpoint -Binding POST -
Protocol SAMLAssertionConsumer -Uri $redirectUrl

Add-ADFSRelyingPartyTrust -Name "ClaimsXray" -Identifier
"urn:microsoft:adfs:claimsxray" -
IssuanceAuthorizationRules $authzRules -
IssuanceTransformRules $issuanceRules -WSFedEndpoint
$redirectUrl -SamlEndpoint $samlEndpoint
```

2. To use OAuth2, we need to create the required OAuth client:

```
Add-AdfsClient -Name "ClaimsXrayClient" -ClientId
"claimsxrayclient" -RedirectUri
https://adfshelp.microsoft.com/ClaimsXray/TokenResponse

if ([System.Environment]::OSVersion.Version.major -gt 6) {
Grant-AdfsApplicationPermission -ServerRoleIdentifier
urn:microsoft:adfs:claimsxray -AllowAllRegisteredClients -
ScopeNames "openid","profile" }
```

3. You will get a **WS-Federation Passive Endpoints** and **SAML Assertion Consumer Endpoint** and OAuth2 configured:



4. Open a browser and test the basic capabilities with one of your test users on the following page: `https://login.inovitdemos.ch/adfs/ls/IdpInitiatedSignon.aspx`.

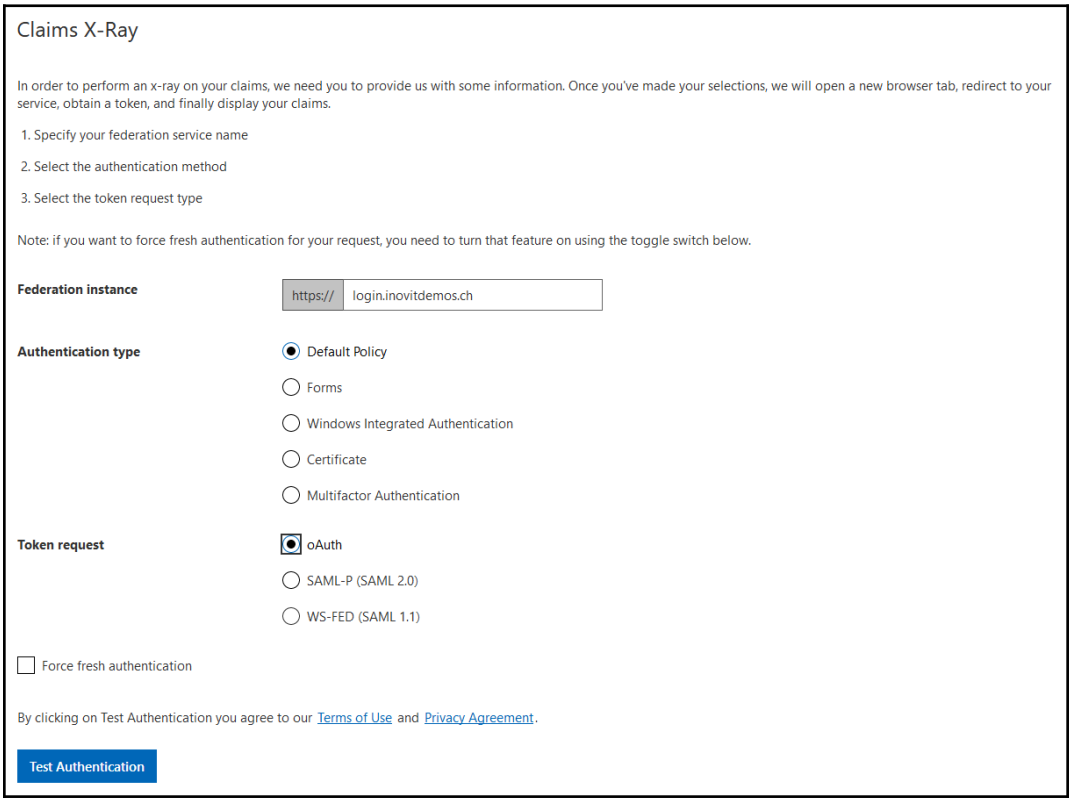
5. Choose ClaimsXRay, log in, and see your token response:

Get your claim information

Now that we have our first helper applications in place, we can start playing around with them. You are already able to view two sample implementations with WS-Federation, SAML, and OAuth.

With the **Claims X-Ray** application, we can run several tests against our ADFS instance. For this, we use source at `https://bit.ly/2EqoPOi`:

1. Let's test our **OAuth** configuration with the following settings:



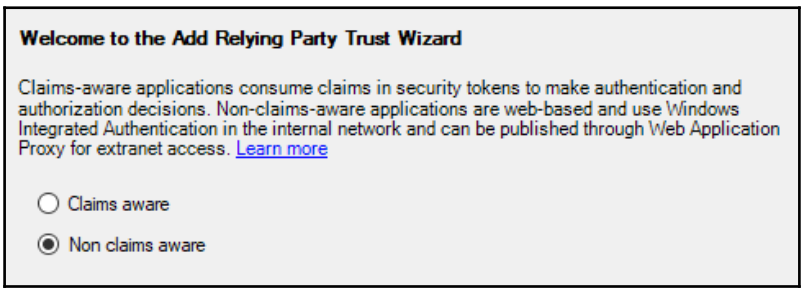
Testing the OAuth configuration with the Claims X-Ray tool

2. Now, you can analyze your response and you will see that you already work with OAuth2:

View the response

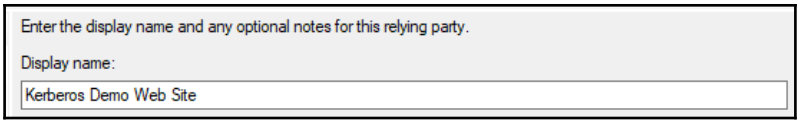
What about if you want to translate a **SAML Token** into a **Kerberos** one:

1. You can work with the **Non claims aware** option in `ADFS`, which appears if you add a new **Relying Party**:



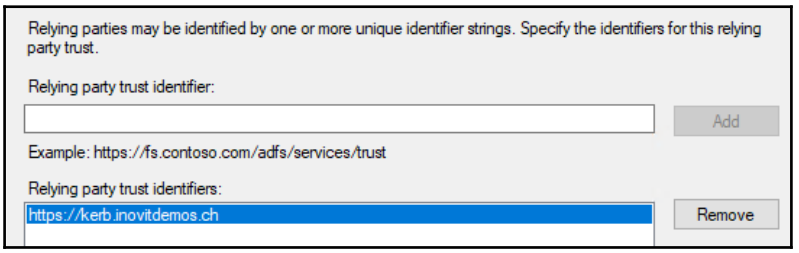
Adding the Kerberos example app

2. Let's try this with our Kerberos example application:



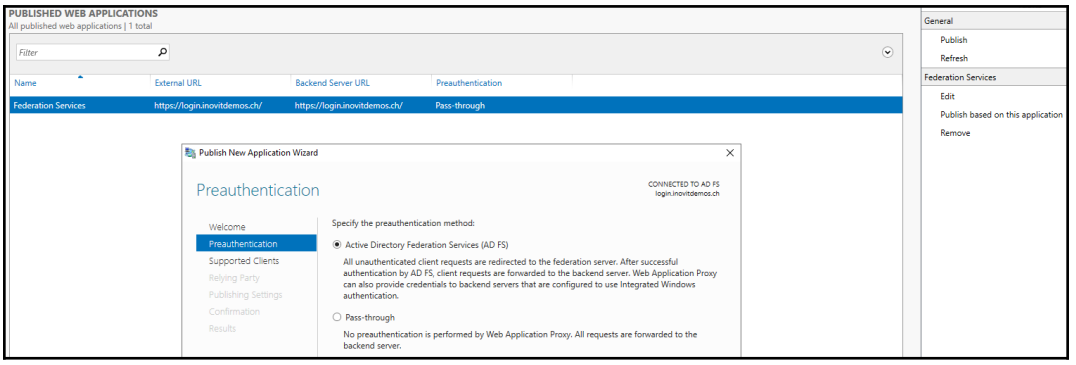
Providing a Display Name

3. Click **Next** and enter `https://kerb.inovitdemos.ch` in **Relying party trust identifier**:



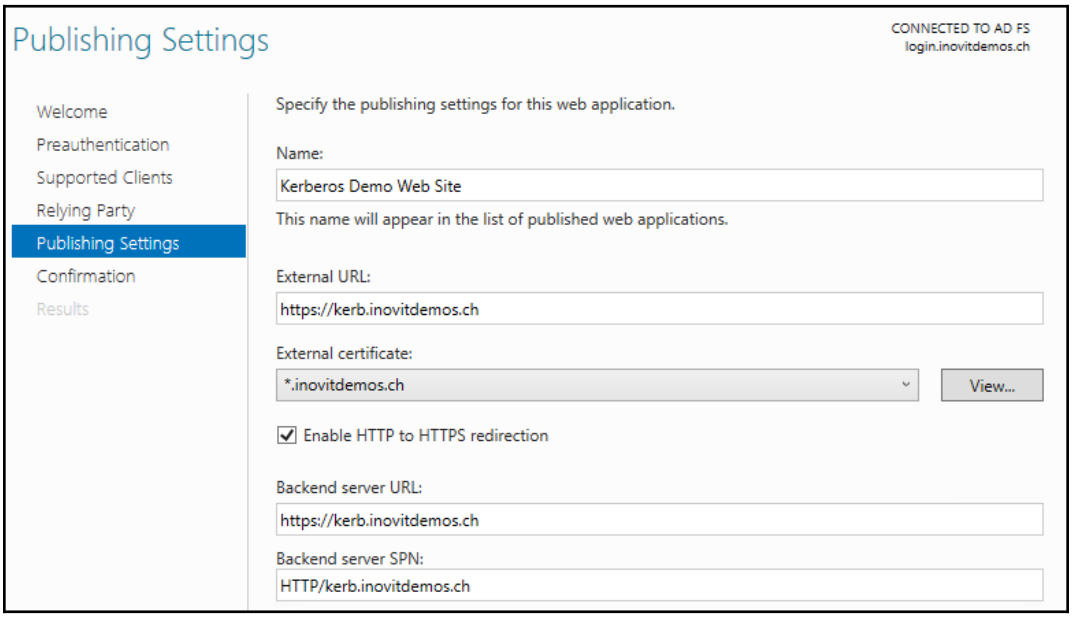
Configuring the identifier

4. Click **Next** in the **Access Control Policy** section and next again until the relying party is added.
5. To test the application, we need to configure the **Web Application Proxy** to publish the app externally.
6. Log in to **YD1URA01** and publish the **Kerberos application** over at the **Remote Access Management** console:



Publishing the Kerberos app with the Web Application Proxy

7. Click **Next** and choose the **Web and MSOFBA** option.
8. Next, choose `Kerberos Demo Web Site` and configure the following settings:

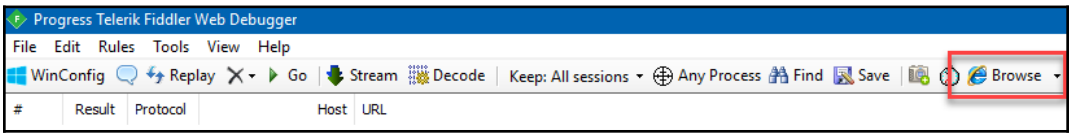


Providing the external/backend server URL and SPN

9. Finish the configuration and jump to your external admin virtual machine or any other external client
10. Open a browser and the Kerberos website, `https://kerb.inovitdemos.ch`, and get a successful login

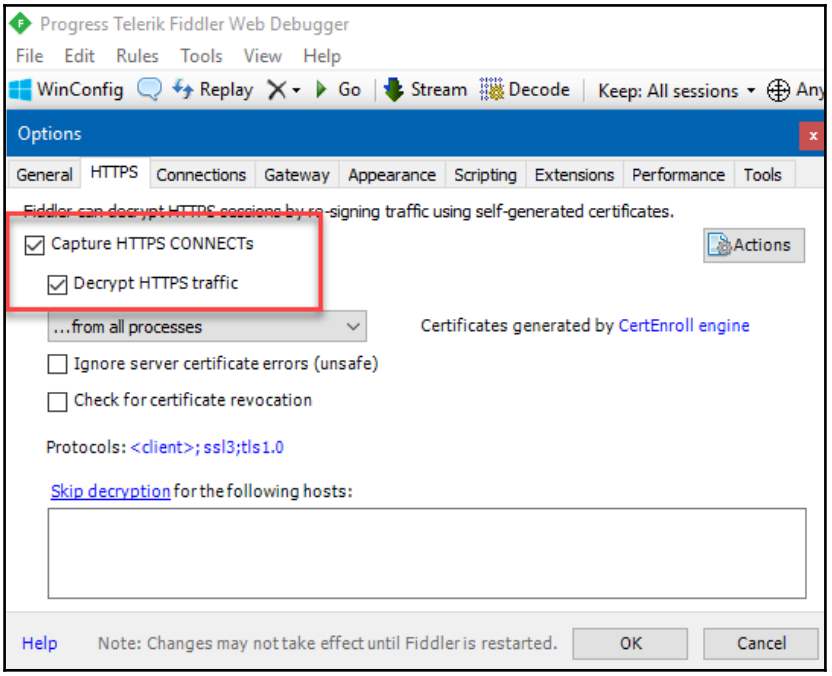
Before we start to test more applications, we will install the Fiddler Debugging tool from `https://bit.ly/2eBzw4C` to analyze our traffic:

1. Start the Fiddler debugging tool:



Using Fiddler for authentication analysis

2. Configure the **HTTPS** capture options:



Capturing HTTPS traffic with Fiddler

3. Press *F12* to start the debugging and open a browser, and you'll see the traffic produced.

Integrating Azure MFA (YD1ADS01)

In this section, we just integrate **Azure MFA** into our **ADFS** farm. We will customize and use this option in `Chapter 8`, *Using Azure AD App Proxy and Web Application Proxy*:

1. First of all, we need to generate a certificate for **Azure MFA** on each server using the following cmdlet:

```
# Replace the tenant ID to your value
$certbase64 = New-AdfsAzureMfaTenantCertificate -TenantID
181031inovitdemos.onmicrosoft.com
```

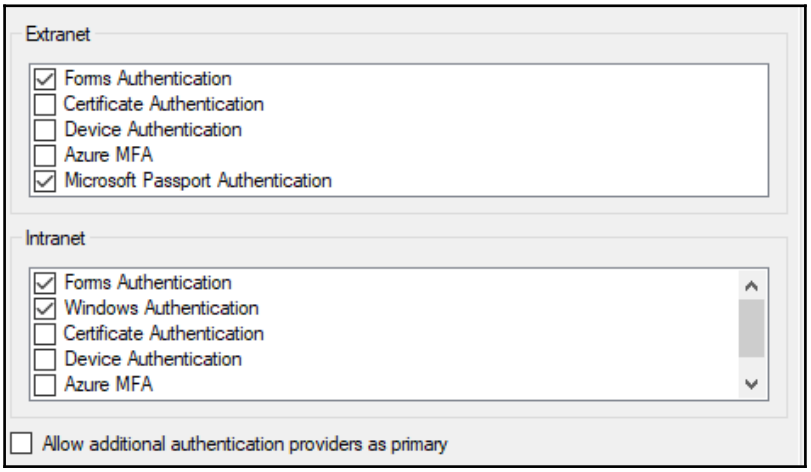
2. Next, we set the certificate as the new credential against the **Azure Multi-Factor Auth** client:

```
# Connect to the MsolService with your global
administrator rights
Connect-MsolService

# Create a new Service Principal Credential the
AppPrincipalId is the hardcoded one for Azure MFA
New-MsolServicePrincipalCredential -AppPrincipalId
981f26a1-7f43-403b-a875-f8b09b8cd720 -Type asymmetric -
Usage verify -Value $certBase64
```

3. Now, we can configure the ADFS farm:

```
Set-AdfsAzureMfaTenant -TenantId
181031inovitdemos.onmicrosoft.com -ClientId
981f26a1-7f43-403b-a875-f8b09b8cd720

Restart-Service adfssrv
```

4. Afterwards, we can see that **Azure MFA** is available as the primary authentication method for intranet/extranet use:



ADFS Primary Authentication configuration

ADFS 2019 allows you to use other third-party authentication providers as the primary method.

With this configuration, we have successful implemented **Azure MFA** in our ADFS farm.

Summary

A lot of practical work, hopefully you liked it! From our perspective, it's always the best to work directly with the authentication methods and capabilities you have just learned. Working through this chapter moves you to touch Azure AD and ADFS to provide several authentication methods. We know that we have many pre-configuration tasks, but it helps you to know the application configurations and to apply the correct authentication configuration. You touched on WS-Federation, SAML, and OAuth2 methods, including Azure AD's user provisioning capabilities.

In the next chapter, we will dive further into the different authentication scenarios and also put the application proxy features in place. We'll be happy to see you in `Chapter 8`, *Using Azure AD App Proxy and Web Application Proxy*.

8
Using the Azure AD App Proxy and the Web Application Proxy

As the successor to `Chapter 7`, *Deploying Solutions on Azure AD and ADFS,* we dig further into the different features of Azure AD and ADFS. We already published the Kerberos on-premises application to external users. We'll do this with another application in this chapter. Furthermore, we will dive into the functionality of the Azure AD app proxy. Additionally, the first usage examples and the basic functionality of conditional access will be shown in this chapter. We do some more claims operation and task customization in `Chapter 10`, *Exploring Azure AD Identity Services*, and provide examples in the code package for the book. With this chapter, you will get all the information for managing the application integration into Azure AD and ADFS. You will also learn to publish applications with the Azure AD or the Web Application Proxy. Additionally, you will be enabled to use conditional access for securing your application access.

This chapter is divided into the following sections:

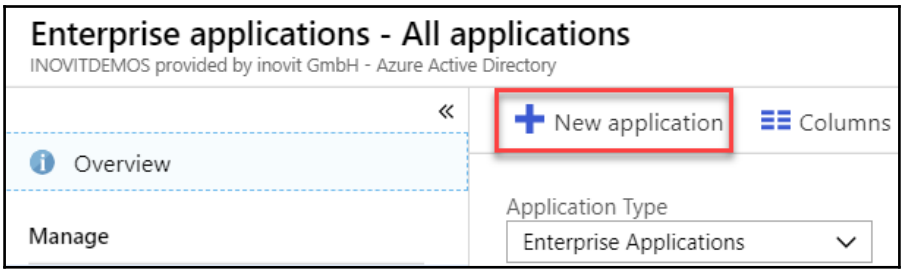
- Configuring additional applications for Azure AD and ADFS
- Publishing with Windows server and Azure AD Web Application Proxy
- Using conditional access

Ready to go? We'll start with the configuration of additional applications for Azure AD and ADFS.

Configuring additional applications for Azure AD and ADFS

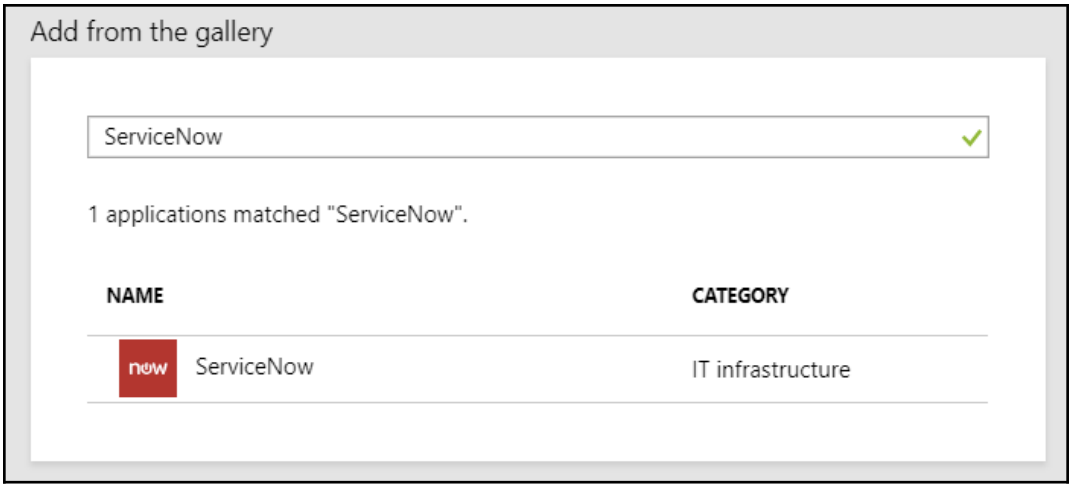
We will continue the deployment and configuration of more business apps to allow you to test the different authentication mechanisms. For our support, we configure **ServiceNow** with **SAML** and active user provisioning from Azure AD to **ServiceNow**:

1. Navigate to **Azure Active Directory** | **Enterprise Applications** and add a **New Application**:



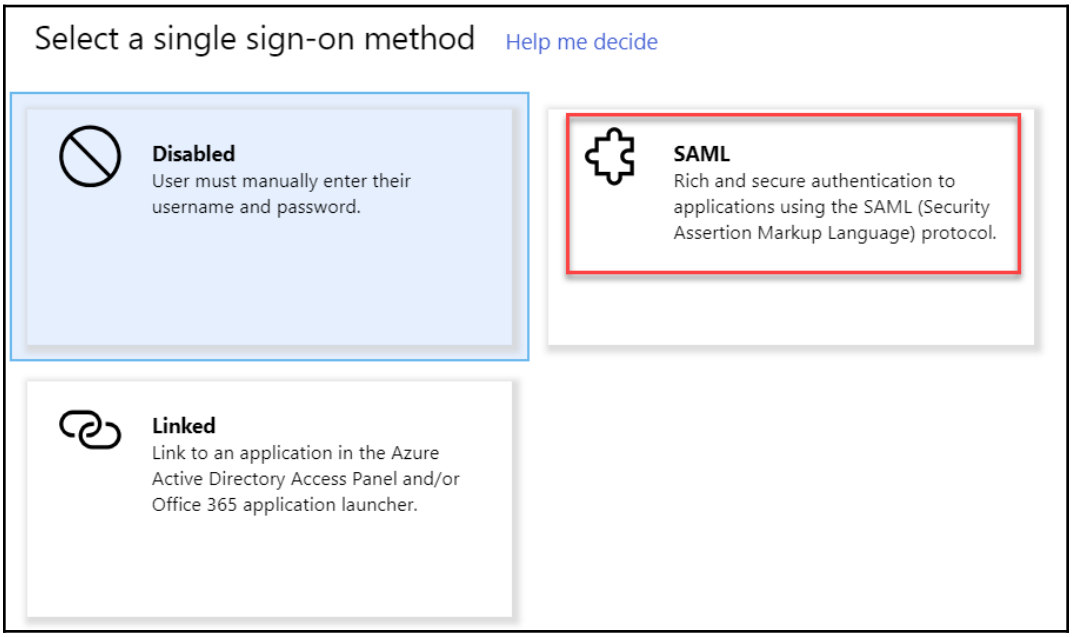
Adding a new Enterprise application

2. Choose **ServiceNow** and add the app from the gallery:



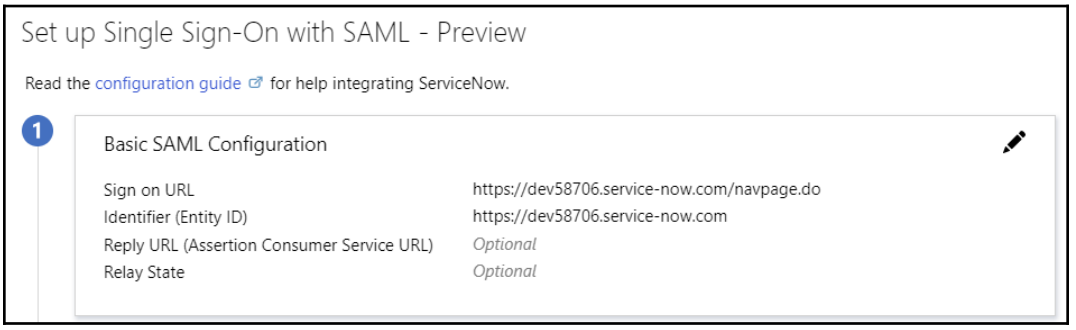
Adding ServiceNow

3. After adding the application, we will configure **SAML** to authenticate our users:



Using SAML as an authentication method

4. In the **Basic SAML Configuration**, we add our **ServiceNow** instance information, as shown in the following screenshot:



Configuring the SAML options

The following URIs need to be used:

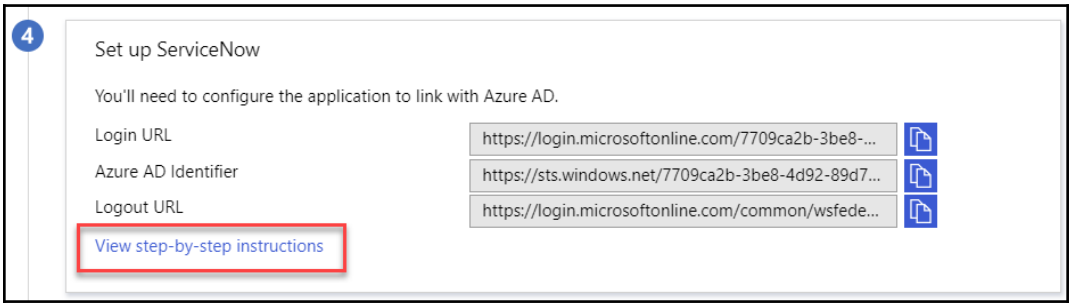
```
https://.service-now.com/navpage.do
https://.service-now.com
```

We will use the automatic configuration method for this chapter.

5. For the manual configuration tasks, you can download the certificate by using the **Base64** option:

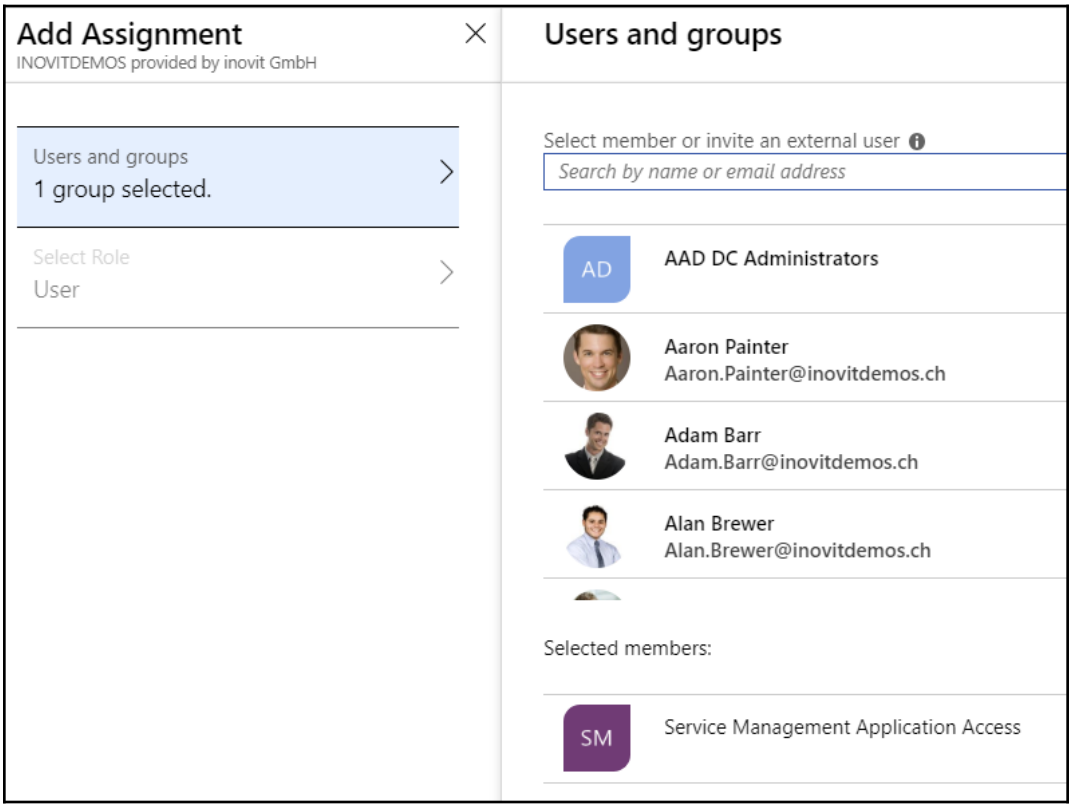
Download the signing certificate

6. Also, for manual configuration tasks, you can copy the following links into a Notepad:



Azure AD endpoints overview

7. Next, we will assign our **Service Management Application Access** group to the application:



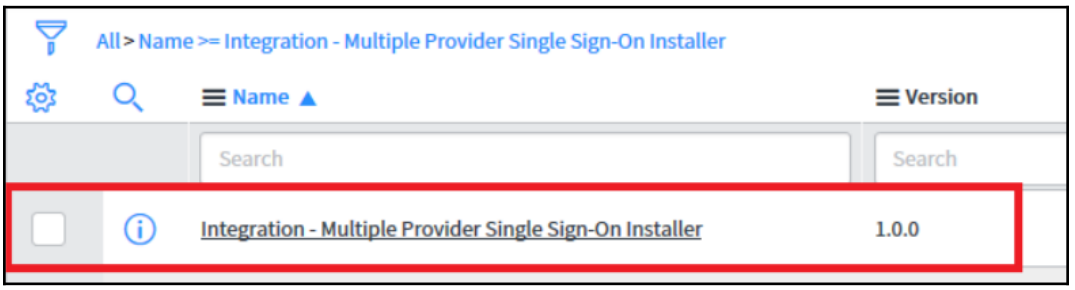
Assigning users to the application

8. Now, we can start to configure the **ServiceNow** instance for Single Sign-On.

9. First, we need to activate the **Multi-Provider Single Sign-On Installer** plugin:

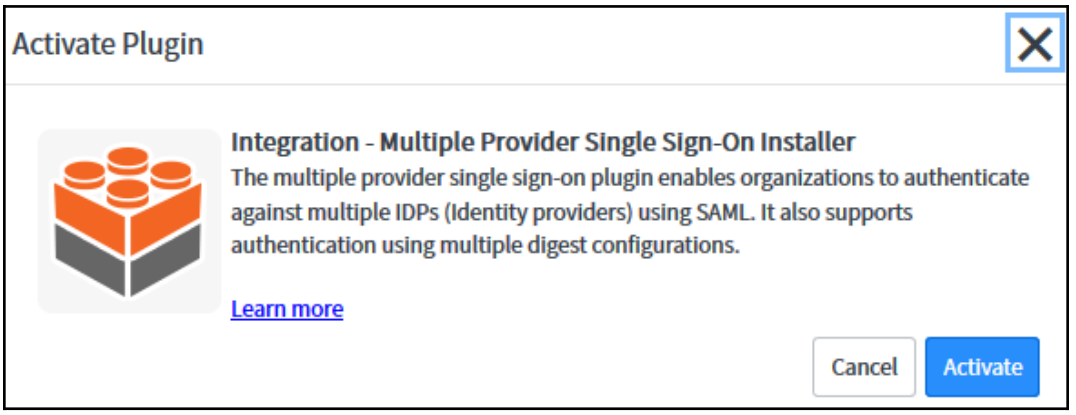
Multi-Provider SSO feature in ServiceNow

10. You will find this option in the navigation pane on the left-hand side: **System Definition** | **Plugins.**
11. Activate the following plugin:



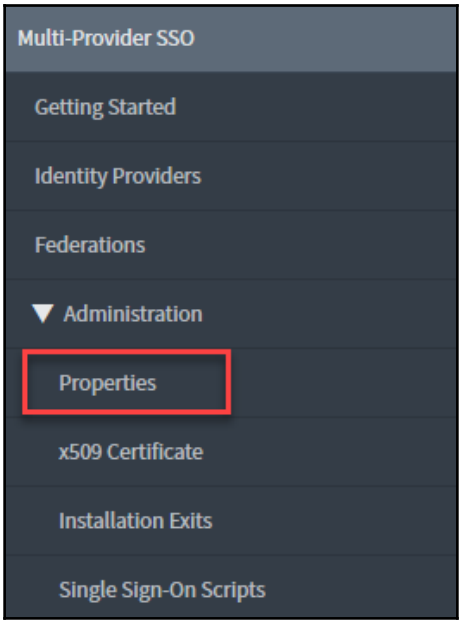
Activating the Multi-Provider SSO feature in ServiceNow

12. Click the **Activate** button in the dialog:



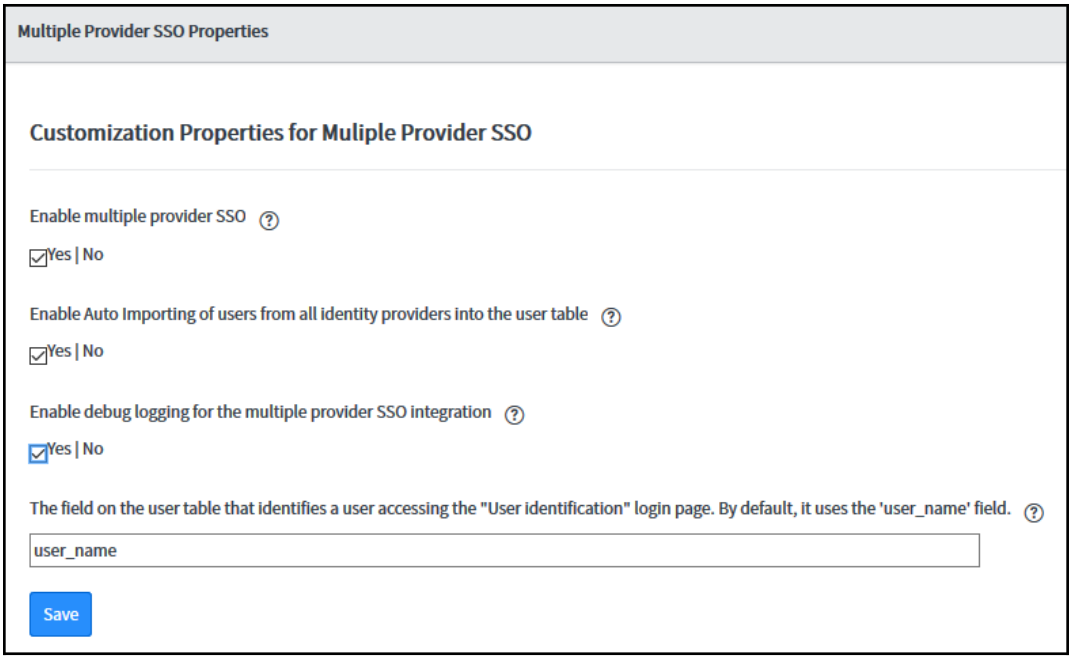
Activation dialog

13. Next, we will configure the plugin under **Multi-Provider SSO** | **Administration** | **Properties**:



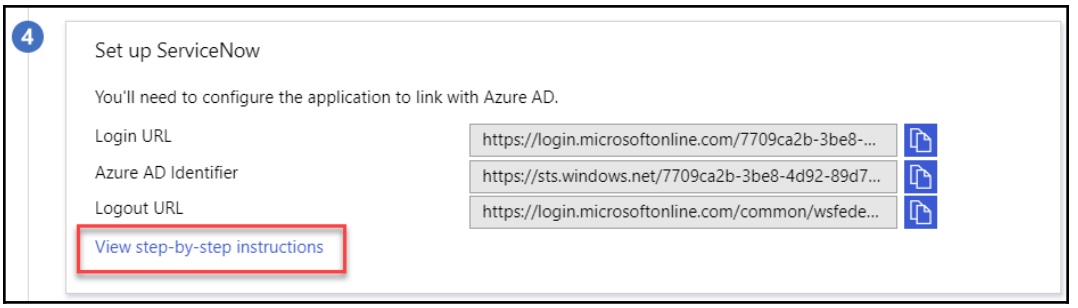
Configuring the newly installed plugin

14. The following features need to be activated:



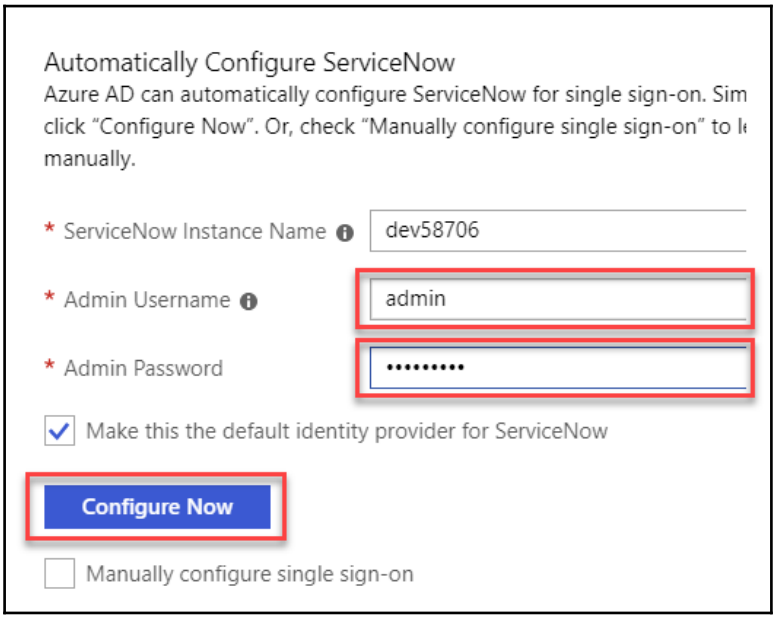
Activating the custom properties for the provider

15. If you want to use the manual configuration method, you need to use the information in your Notepad.
16. We use the automatic option for the configuration; it's a little bit of a hidden option!
17. You will find it under **View step-by-step instructions** as shown in the following screenshot, in the SAML configuration section:



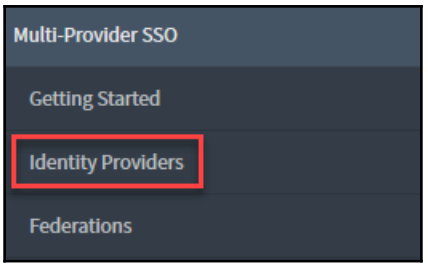
Automatic configuration option for the ServiceNow SSO Provider

18. Click the option and the following dialog appears; enter your **ServiceNow** administrative credentials:



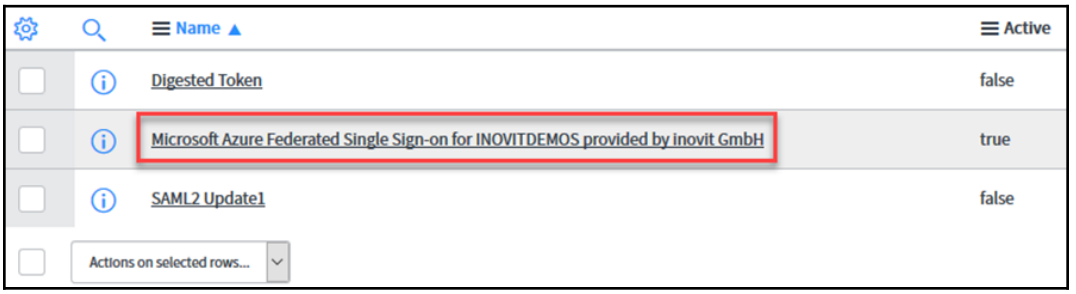
Configuring the SAML connection

19. Click **Configure Now**.
20. Next, we can do additional configuration tasks on the **ServiceNow** part.
21. Navigate to **Multi-Provider SSO** | **Identity Providers**:



Identity Provider configuration

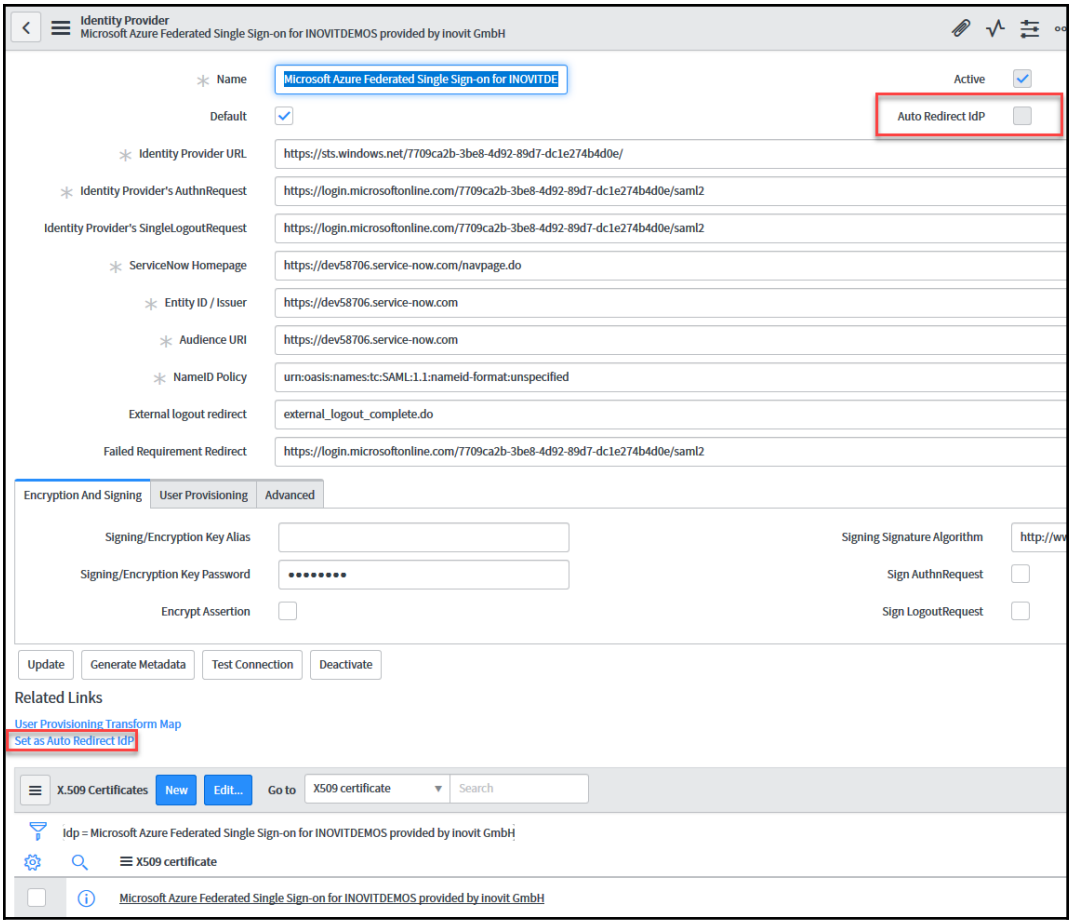
22. You will find the automatically generated **Identity Provider**:



Start configuring the newly created Azure AD connection in ServiceNow

23. We need to remove the populated Identity Provider's `SingleLogoutRequest` URL.

24. All the other fields can be left as their defaults:

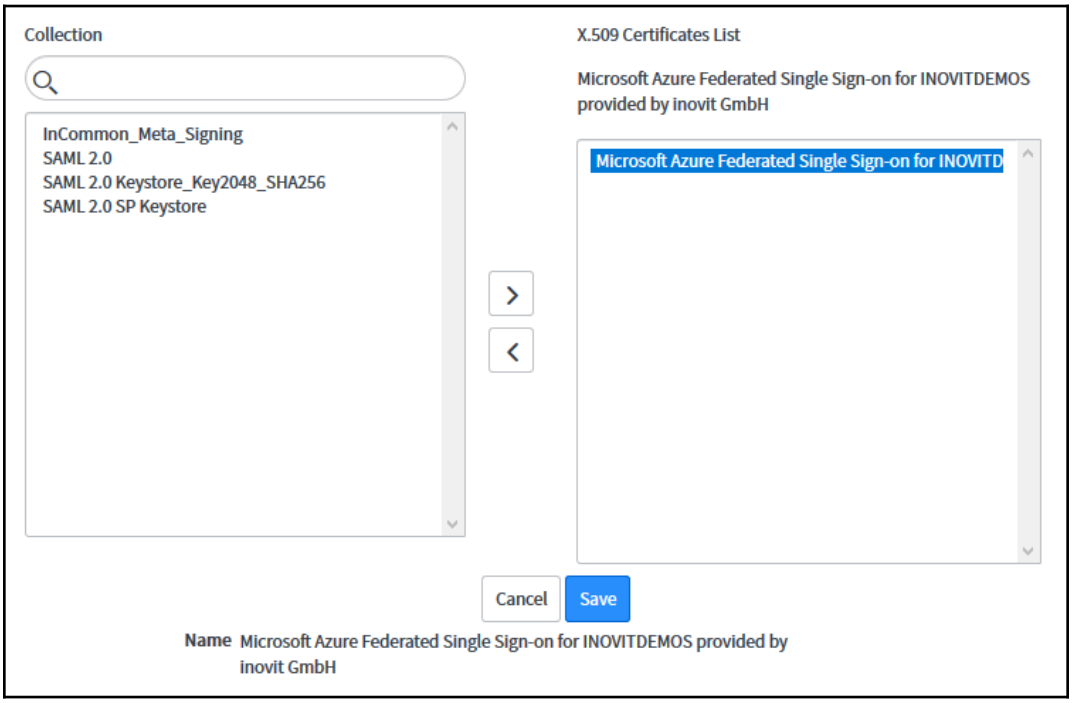


Configuring the IdP including the Auto Redirect option

25. Very important! Set the **Auto Redirect IdP** option, with the expected result:

Auto Redirect IdP option

26. Next, we need to choose the correct certificate for our configuration by navigating to the **X.509** section:



Configuring the correct certificate

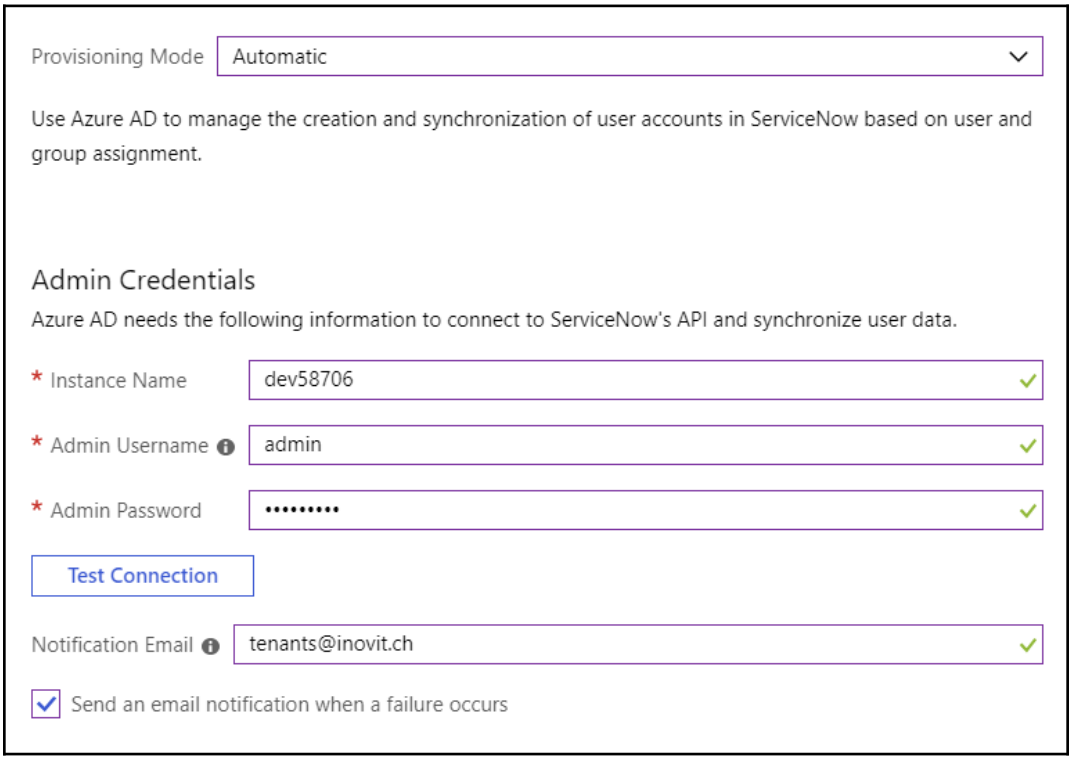
27. **Save** the configuration.

Now, we have finished the **SAML** configuration of **SeviceNow** and we will follow up with the user provisioning:

1. Navigate to the **Provisioning** section of the **ServiceNow** application and change the **Provisioning Mode** to **Automatic.**
2. Insert the following values:

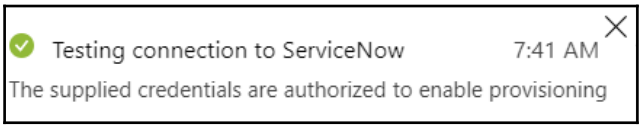
    - **Instance Name**
    - **Admin Username**
    - **Admin Password**

- **Notification Email** (and check **Send an email notification when a failure occurs**):



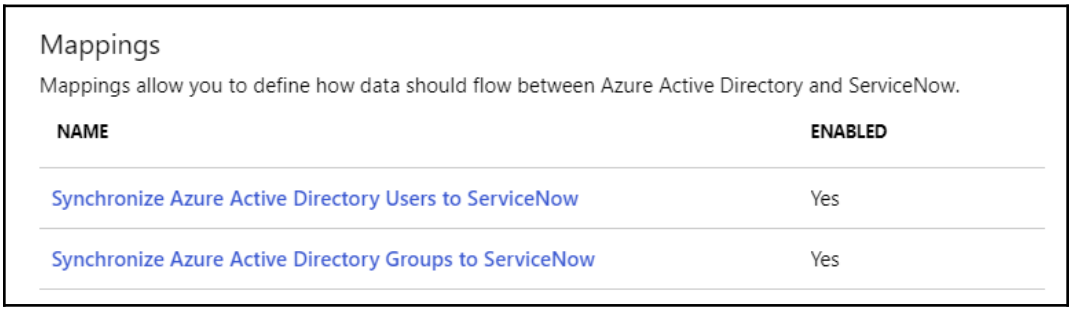
Configuring the SCIM provisioning feature

3. Click **Test Connection** and you should receive a notification like the following example. The test of the connection is required to save:



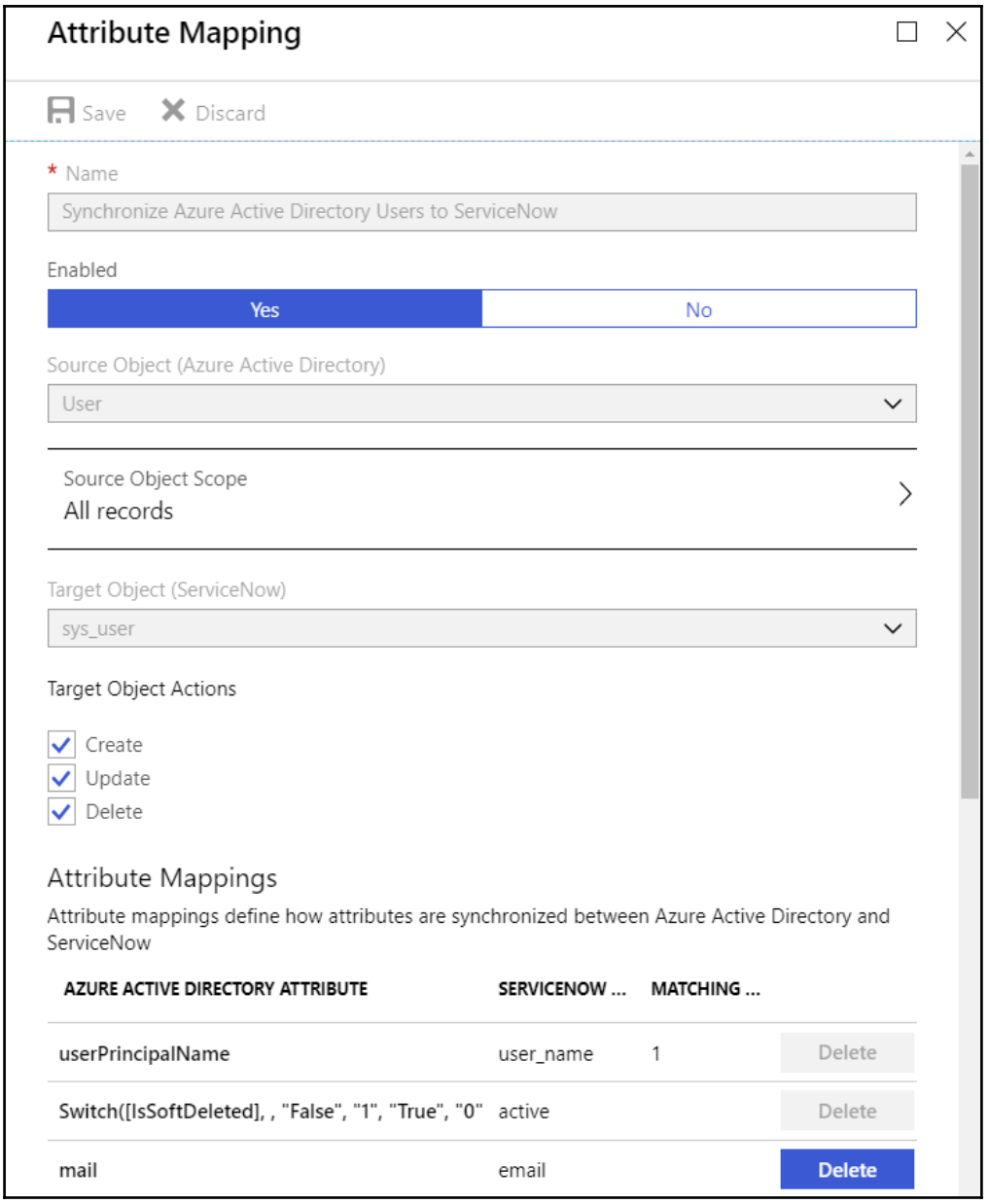
Connection Test

4. Next, we will take a look into the **synchronization configuration** for users and groups.
5. Click the **Synchronize Azure Active Directory Users to ServiceNow** rule:



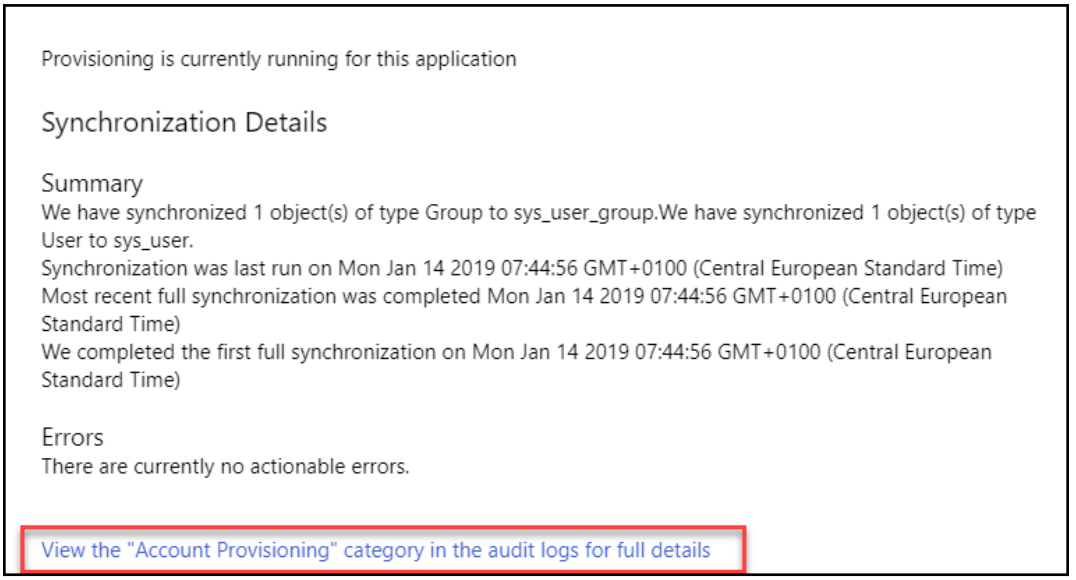
Configuring the Synchronization Rules

6. You will find the active **Attribute Mapping**. If you need to handle special considerations, such as if you connect to a filled **ServiceNow** instance, you can modify the mappings to fit your needs:



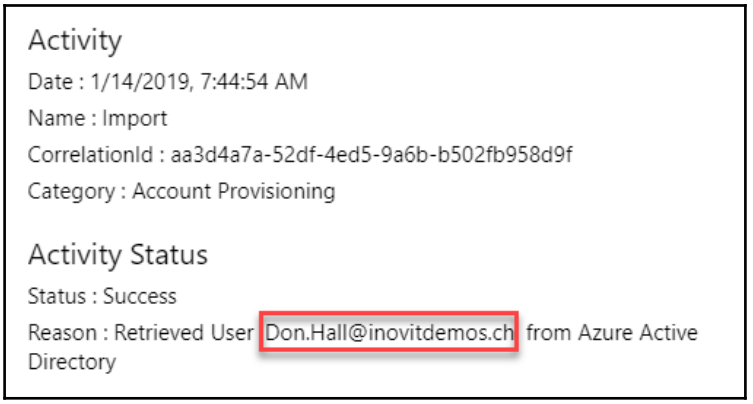
Synchronization Rule Settings

7. You will also get an actual overview of the **Synchronization Details**:



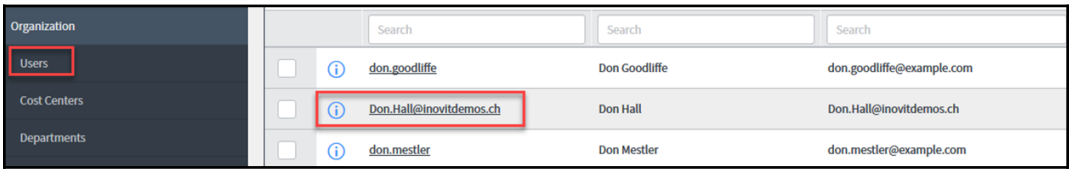
Synchronization details for monitoring and auditing

8. By clicking on **View the "Account Provisioning" category in the audit logs for full details** you will get the following screenshot:



Actual state for the test user

9. Next, we will change to the **Organization** | **User** section in your **ServiceNow** instance.
10. Search for your test user, in my case `Don.Hall@inovitdemos.ch`, and you will see the result of the automatic provisioning:



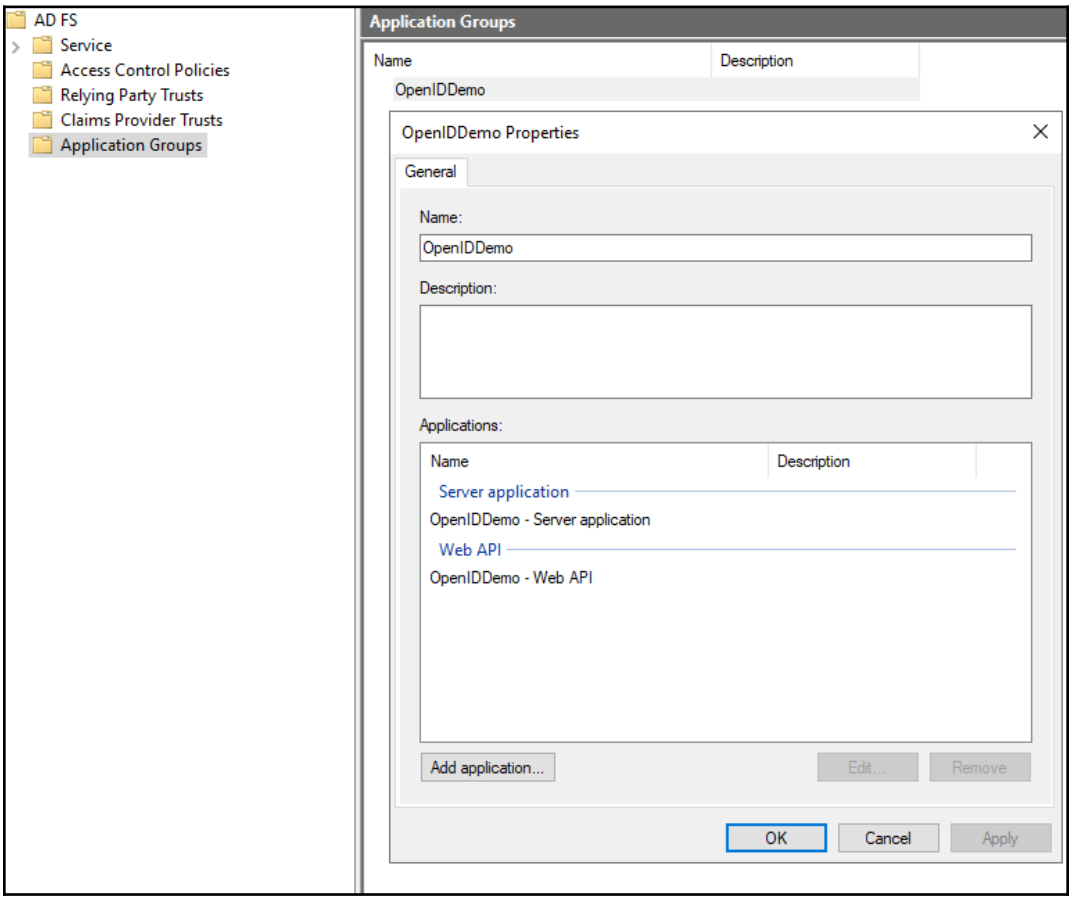
Verifying the synchronized test user in ServiceNow

The initial synchronization takes longer to perform than subsequent synchronizations, which occur approximately every **40 minutes** as long as the service is running.

The next scenario we want to accomplish is the use of **OpenID Connect** and your on-premises ADFS infrastructure. To follow the authentication flow, use `Chapter 6`, *Managing Authentication Protocols*.

To configure this scenario on your **YD1APP01** server, we need to copy the `active-directory-dotnet-webapp-openidconnect` folder from the code package to your desktop.

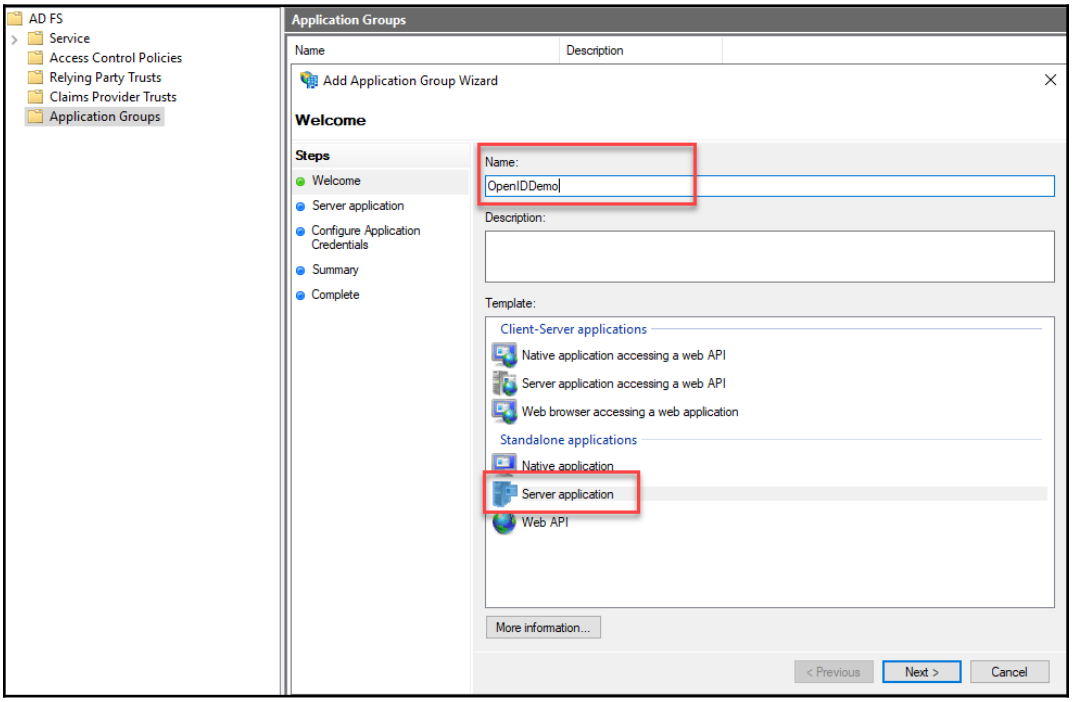
We configure your ADFS server (**YD1ADS01**) on your domain controller to get the following expected result at the end, a full working OpenID authenticated application:



Result of the following configuration steps

In the next steps, we will configure the scenario:

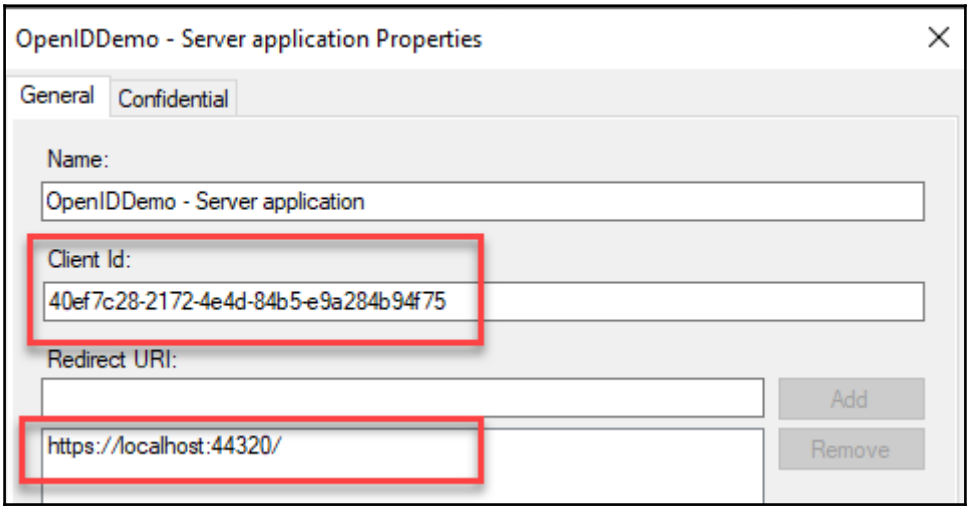
1. We start with the creation of a new application group including a **Server application** (standalone application):



Configuring the new application group

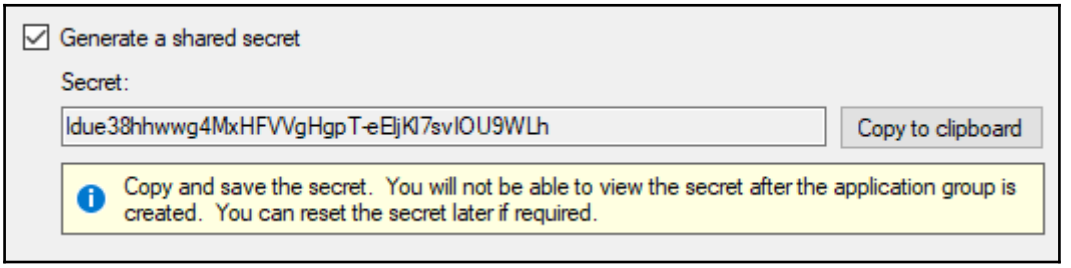
2. Next, a **Client Id** is generated; copy the ID to a notepad.

3. Enter the redirect URI, `https://localhost:44320/`, under which the application will run:



Providing the application properties

4. Click **Next** and check **Generate a shared secret box**; copy the key to the notepad:



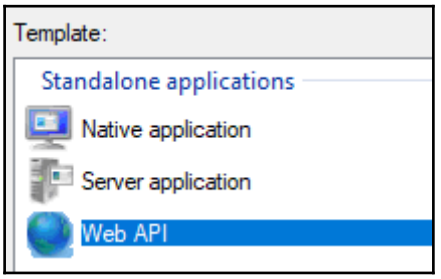
Generating the shared key

5. Complete the wizard.
6. Next, we will create the Web API configuration by double-clicking on your new application group.

7. Choose **Add application**:

Web API configuration

8. Use the **Web API** option:



Choosing the Web API option

9. Next, add the Identifier for the API: `https://inovitdemos.ch/OpenIDDemo`.
10. Click **Next** in the **Access Control Policy** section and all the other sections to finish the wizard.
11. For now, we change to the **YD1APP01** server and open our Visual Studio.
12. Open the Solution file of the demo code: `WebApp-OpenIDConnect-DotNet.sln`.
13. Navigate in Solution Explorer to the `web.config` file: `WebApp-OpenIDConnect-DotNet.sln`.
14. Modify the code so that it looks like the following example:
    - Copy your **Client Id** from your notepad.
    - Configure the `ida:ADFSDiscoveryDoc` value so that it contains your ADFS FQDN: `https://login.inovitdemos.ch`.
    - Verify that the `ida:PostLogoutRedirectUri` value contains `https://localhost:44320/`:

```
<appSettings>
<add key="webpages:Version" value="3.0.0.0" />
<add key="webpages:Enabled" value="false" />
<add key="ClientValidationEnabled" value="true" />
```

```
 <add key="UnobtrusiveJavaScriptEnabled" value="true" />
 <add key="ida:ClientId" value="40ef7c28-2172-4e4d-84b5-
e9a284b94f75" />
 <add key="ida:ADFSDiscoveryDoc"
value="https://login.inovitdemos.ch/adfs/.well-known/openid-
configuration" />
 <add key="ida:PostLogoutRedirectUri"
value="https://localhost:44320/" />
</appSettings>
```

15. Next, navigate to the *App_Start* section and the `Startup.Auth.cs` file.
16. Directly under the public partial class `Startup`, the code needs to look like the following to tweak the OpenID Connect middleware initialization logic:

```
private static string clientId =
ConfigurationManager.AppSettings["ida:ClientId"];
 //private static string aadInstance =
ConfigurationManager.AppSettings["ida:AADInstance"];
 //private static string tenant =
ConfigurationManager.AppSettings["ida:Tenant"];
private static string metadataAddress =
ConfigurationManager.AppSettings["ida:ADFSDiscoveryDoc"];
private static string postLogoutRedirectUri =
ConfigurationManager.AppSettings["ida:PostLogoutRedirectUri"];
 //string authority =
String.Format(CultureInfo.InvariantCulture, aadInstance,
tenant);
```

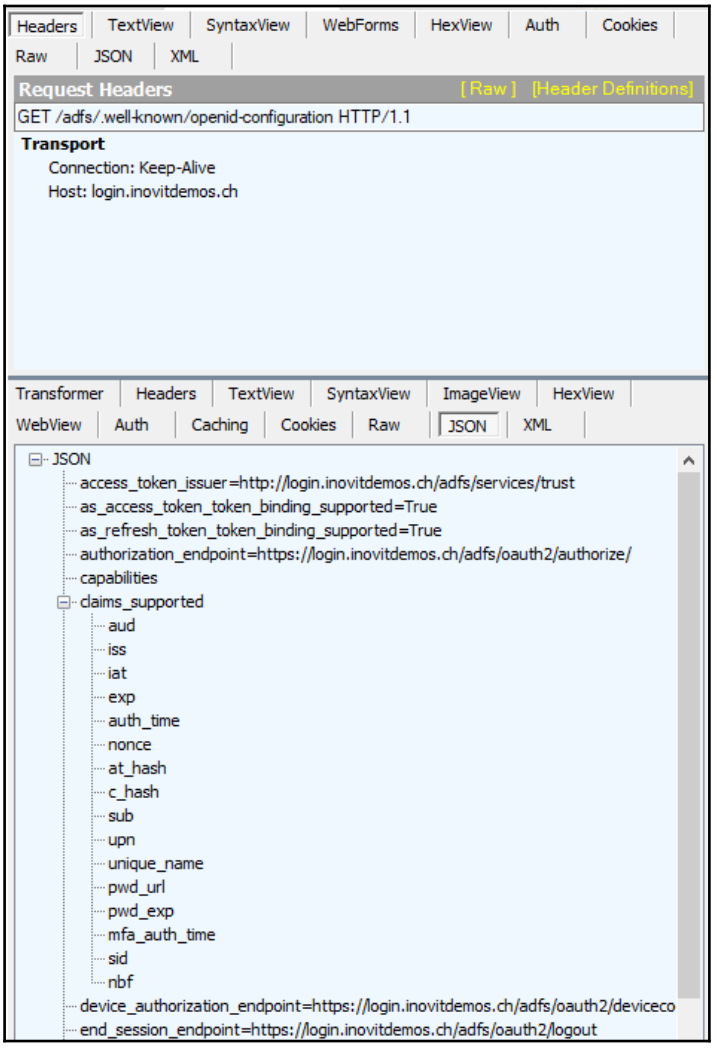
We don't need the Azure AD configuration, so we comment this part out!

17. Next, we need to modify the OpenID Connect middleware options:

```
app.UseOpenIdConnectAuthentication(
new OpenIdConnectAuthenticationOptions
 {
 ClientId = clientId,
 //Authority = authority,
 MetadataAddress = metadataAddress,
 RedirectUri = postLogoutRedirectUri,
 PostLogoutRedirectUri = postLogoutRedirectUri
```

ADFS does enforce the `redirect_uri` in the request. Azure AD doesn't do that.

18. Now, we can verify the app; press *F5* in Visual Studio. To analyze the traffic, you can use Fiddler. Note that you will be stuck with Fiddler on the authentication part:



Fiddler result of the authentication

19. Well done, you have configured an application that uses OpenID Connect!

Publishing with Windows server and Azure AD Web Application Proxy

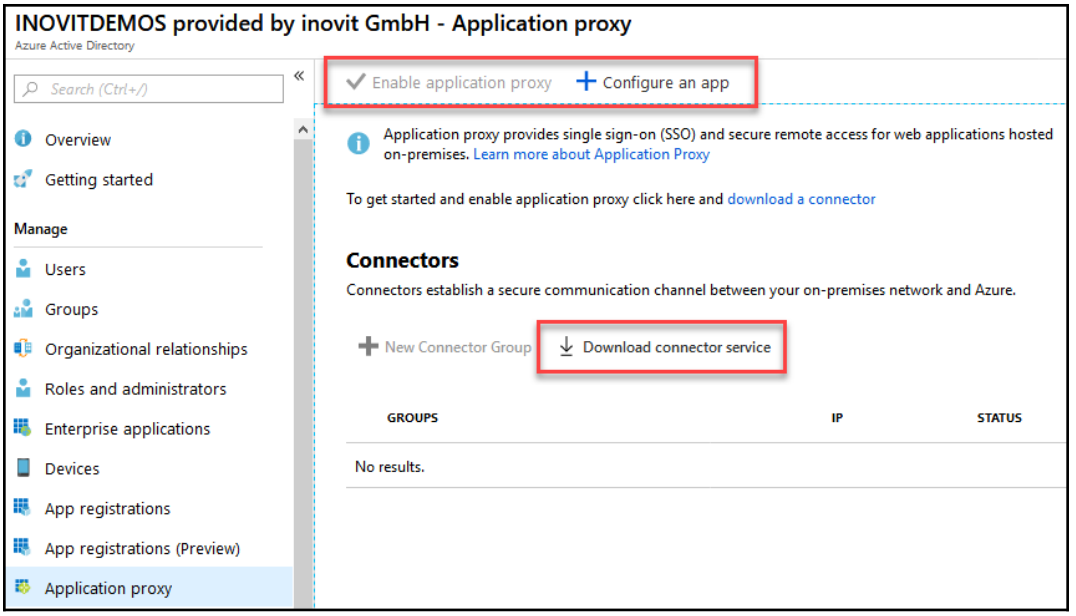
The Azure AD Application Proxy is similar to the on-premises Web Application Proxy role, starting in Windows Server 2012 R2. With this service, you can enable external access for on-premises applications. Azure AD Application Proxy requires an Azure AD Basic or an Azure AD Premium subscription. The connection is made directly with Azure and done through a proxy into the private network, with an application proxy agent installed on the on-premises web application server.

Let's run a very common use case to include a Kerberos on-premises application into our Azure AD Access UI, `https://myapps.microsoft.com`. We use our existing application to configure the scenario:

1. Log in to `https://portal.azure.com` and choose the Azure Active Directory blade.
2. Under Application proxy, we first need to download and install the application proxy agent on our **YD1APP01** server.

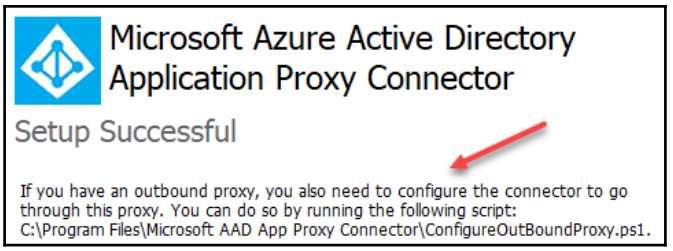
You don't need to install the agent directly on the application server. It's also possible to use any other server, or also additional agents for redundancy requirements. The agent needs to be installed on a server in the same or a correctly trusted domain/forest to support Kerberos constrained delegation, and the SPN of the application needs to be done on every agent instance. The domain function level needs to be Windows Server 2012 at a minimum. Also, the correct ports need to be accessible from the other server.

3. Download the agent onto the server and start the installation:



Configure the Azure AD Application Proxy

4. Agree to the license terms and conditions and click **Next.**
5. Next, you need to provide global administrator credentials to register the agent to Azure AD.
6. Recognize the notification for outbound proxy usage:

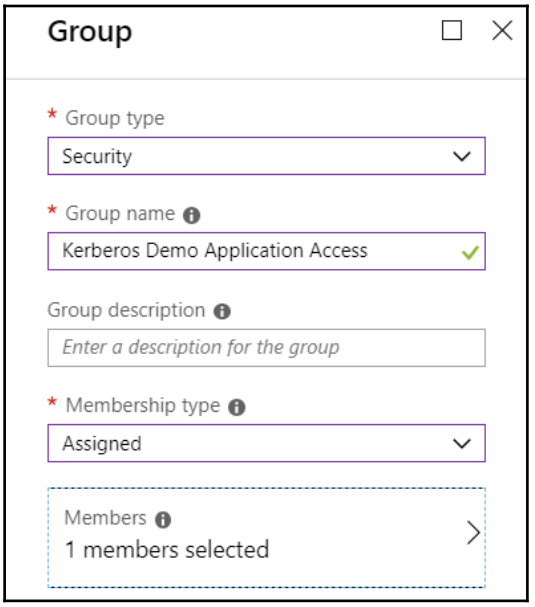


Installation of the Application Proxy Connector

7. The expected result is an **Active** agent:

Active agents overview

8. Now, we can start to configure our Kerberos-based on-premises application.
9. First, we create a group that can be assigned to grant access to the portal:

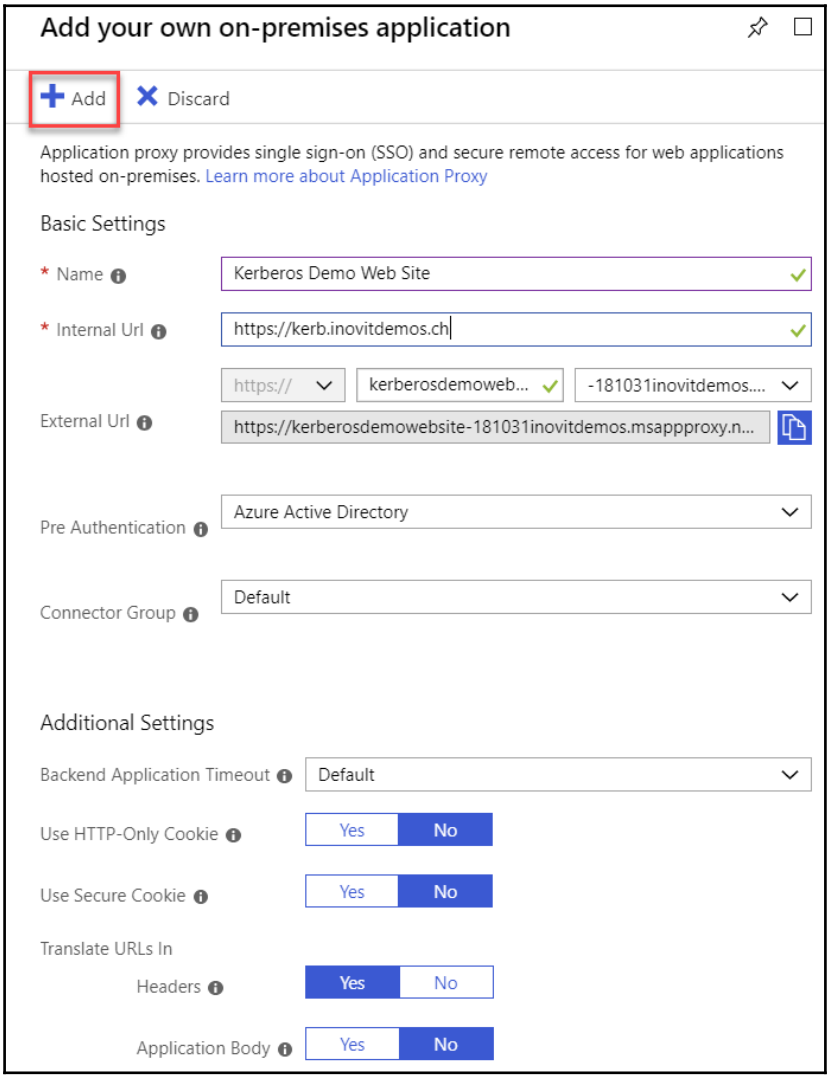


Assigning the needed test group

10. Assign one test user to your group.

In this example, we use the option that we don't publish the application with an official public FQDN. In that case, we don't need to open any firewall ports and the traffic to the app is working through the connector.

11. Next, we will configure the app proxy configuration:

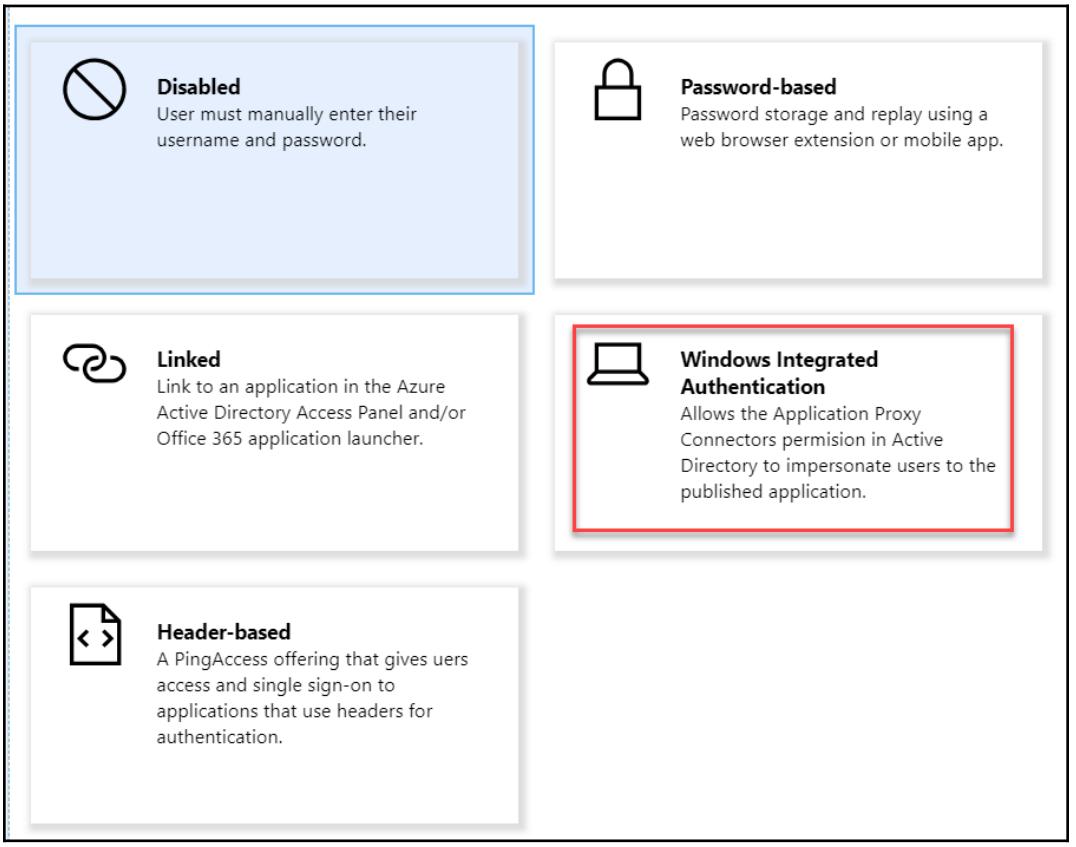


Configuring the on-premises application properties

12. As you can see, you are able to provide several additional settings to address your needs.

You can find more information about the different options at the following reference: `https://docs.microsoft.com/en-us/azure/active-directory/manage-apps/application-proxy-add-on-premises-application`.

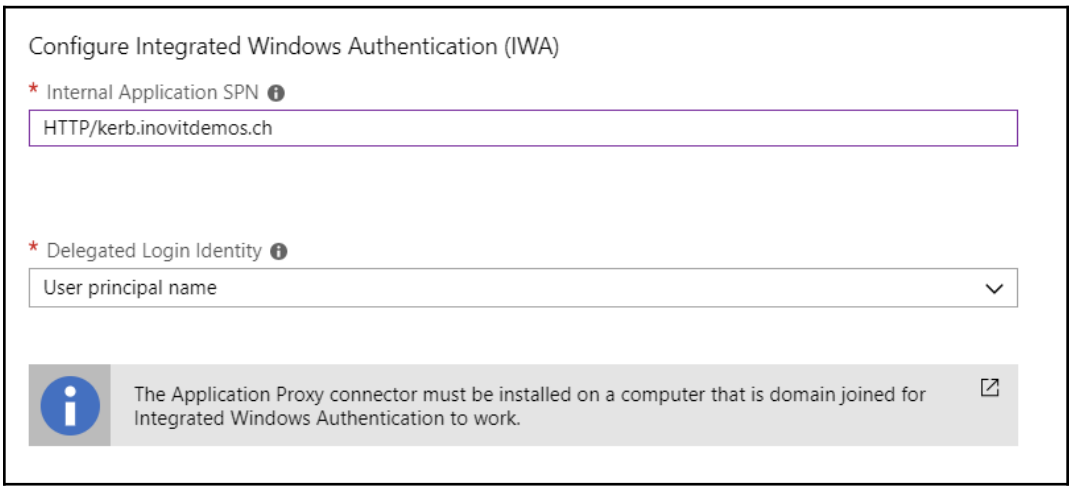
13. In the next step, we will configure the **Single-Sign-On** options for our application.
14. Choose the **Windows Integrated Authentication** option:



Choosing the Windows Integrated Authentication option

If you need to use Header-based authentication, the integration with PingAccess will be your friend. You can find more information about this integration at the following source: `https://docs.microsoft.com/en-us/azure/active-directory/manage-apps/application-proxy-configure-single-sign-on-with-ping-access`.

15. Enter your **Internal Application SPN** to configure the Kerberos part.
16. Keep in mind that only Kerberos Constrained Delegation will work:



Configure the SPN for the App

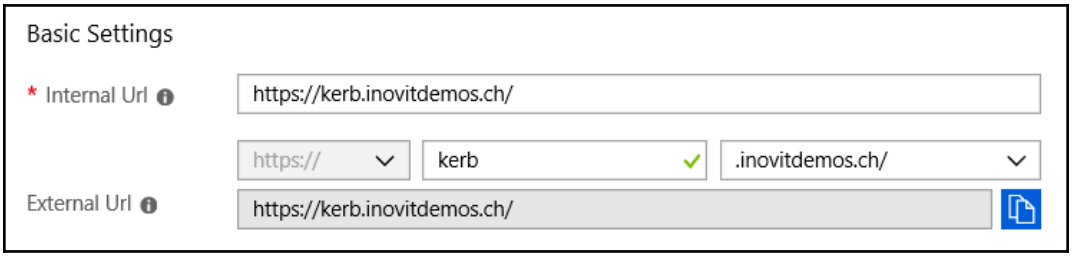
17. After this configuration, we can test the application with the assigned user in your access group, and you should get a similar result to this:

Authentication Method	**Negotiate (KERBEROS)**	Request.ServerVariables("AUTH_TYPE")
Identity	**INOVITDEMOS\donh**	Request.ServerVariables("AUTH_USER") or System.Threading.Thread.CurrentPrincipal.Identity
Windows identity	**INOVITDEMOS\svckrbapp**	System.Security.Principal.WindowsIdentity.Getcurrent

Result of a successful Kerberos authentication on the example app

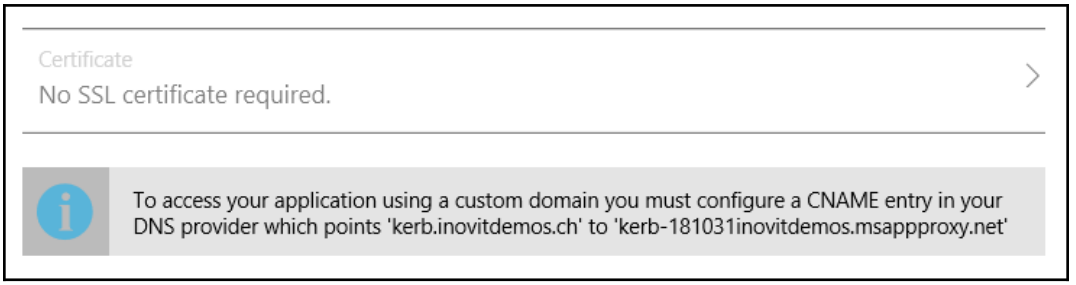
This publishing method also allows us to use a public FQDN. This option can be useful if your application sends notification mails with links inside. To realize this use case, you need to change the external URL and provide a public SSL certificate. You can use your wildcard certificate or issue a separate SSL certificate with Let's encrypt.

18. You will find the settings under **Basic Settings**:



Defining the basic app settings

For the certificate, you need to use the following section. Don't forget to create the **CNAME** inside your public **DNS**:



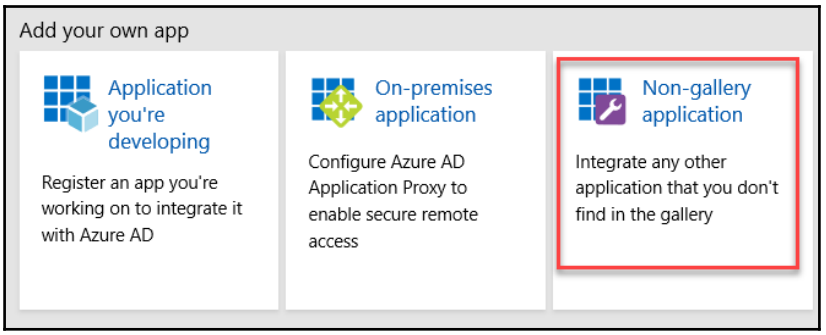
View the DNS configuration options

The next authentication method we can use for cloud or on-premises applications (through Azure AD Web Application Proxy) is the password-based option. We use it for form-based authentication activated applications that use their own identity provider.

Your credentials will be securely stored under the user or group object.

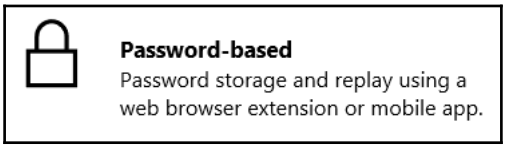
In the next steps, we will add a password-based application access:

1. Add a new application and use the **Non-gallery application** option.
2. Choose the **Password-based** option under the **Single sign-on** method:



Choosing a non-gallery application

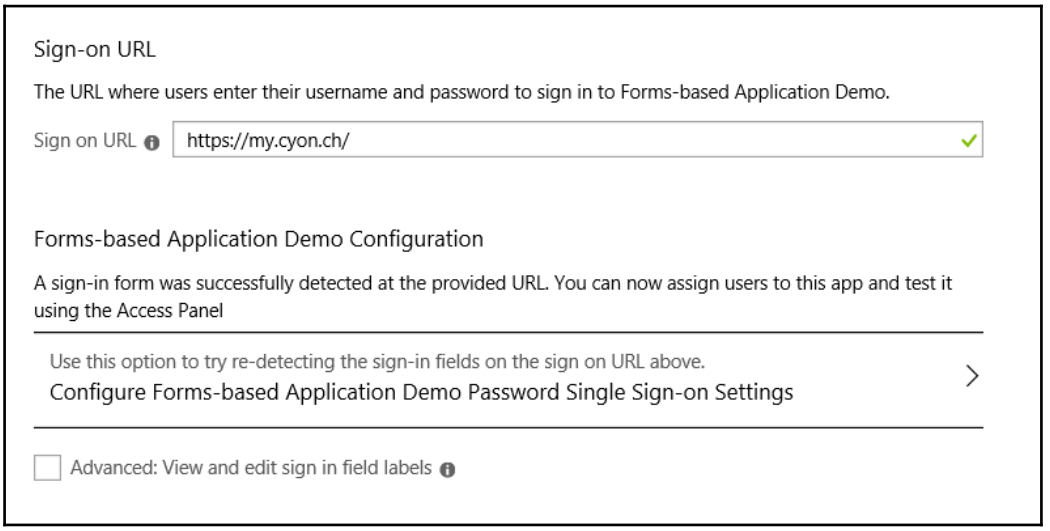
3. Provide a name for your application and navigate to the **Single-Sign-On** configuration:



Using the password-based authentication method

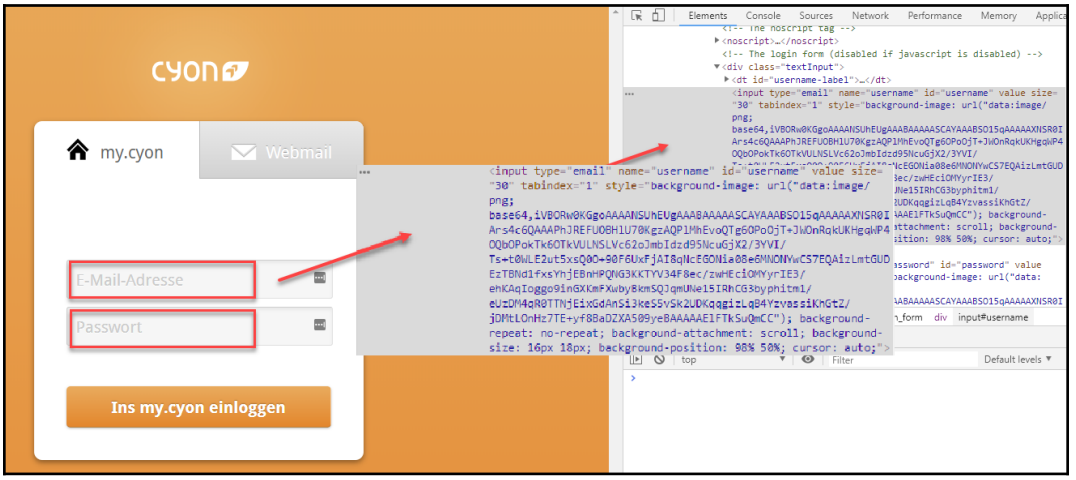
4. Provide the **Sign-On URL** and detect the sign-in fields of your application.
5. There are three options to do this configuration:
    - Automatically
    - Manually

- With custom configuration in the Advanced View:



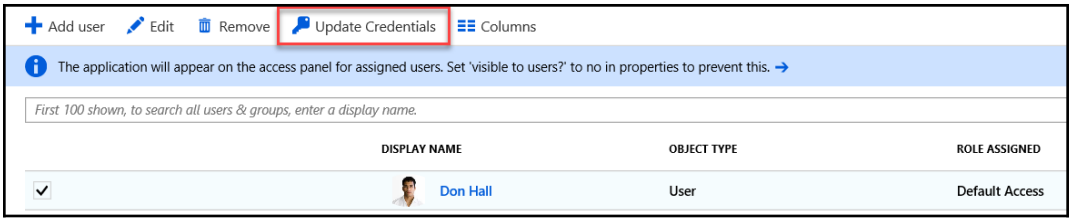
Detecting the login fields of the application

6. In this example, we used our public DNS provider's login page.
7. You can view the field labels with any browser and the developer tools to provide the values if the Azure AD mechanism doesn't get the correct values:



Analyzing the code to find the login fields/labels

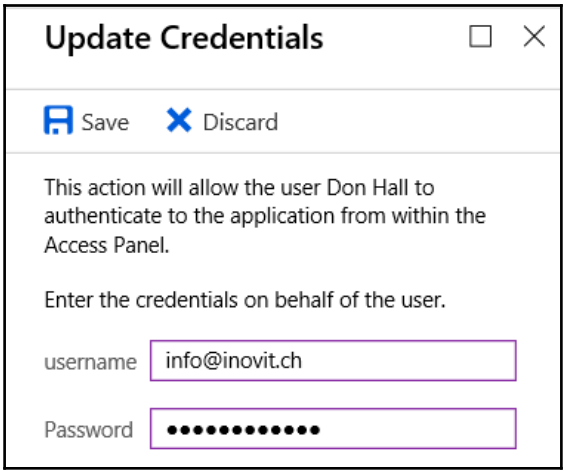
8. Next, you need to assign the user or group to the application:



Updating the credentials on the user/group

You have the option to provide the credentials hidden to the user or group, and the option that the credentials will be saved with the first access.

9. For now, provide the credentials with the **Update Credentials** button:



Update credentials dialog

10. Now, you are ready to test the application with your test user.

Keep in mind that the user needs to have the Access Panel UI extension installed or he will get asked to install it. You can also deploy this extension to all your computers using a group policy or any other software deployment tool. Find out more information about this at `https://docs.microsoft.com/en-us/azure/active-directory/manage-apps/access-panel-extension-problem-installing`

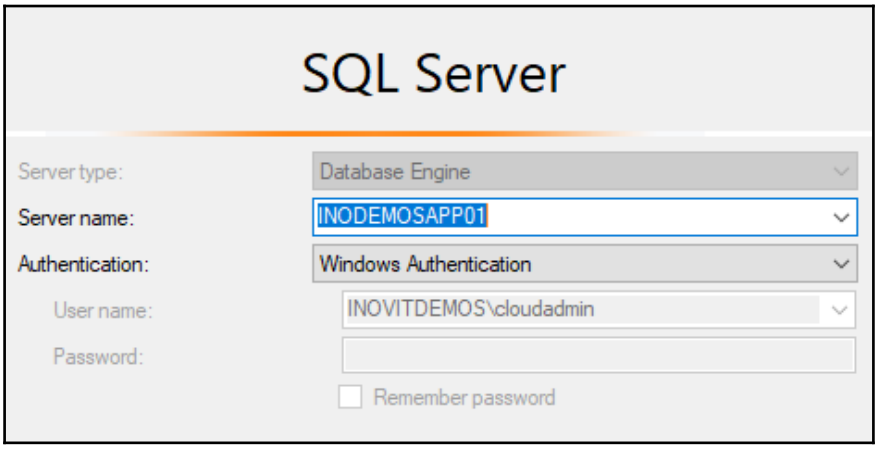
11. To test the same capabilities with your on-premises infrastructure, we provide you with a small challenge. If you don't get it running, send an email to `support@inovit.ch` and we will be happy to provide you with the answer.
12. You need to deploy the test application with the following steps on the **YD1APP01 server**.
13. Create a DNS entry for the app on your domain controller (**YD1ADS01**), `forms.inovitdemos.ch`:

```
Add-DnsServerResourceRecord -ZoneName "inovitdemos.ch" -A -
Name "forms" -IPv4Address "10.0.0.6"
```

14. Create the service account under which the application runs on your app server:

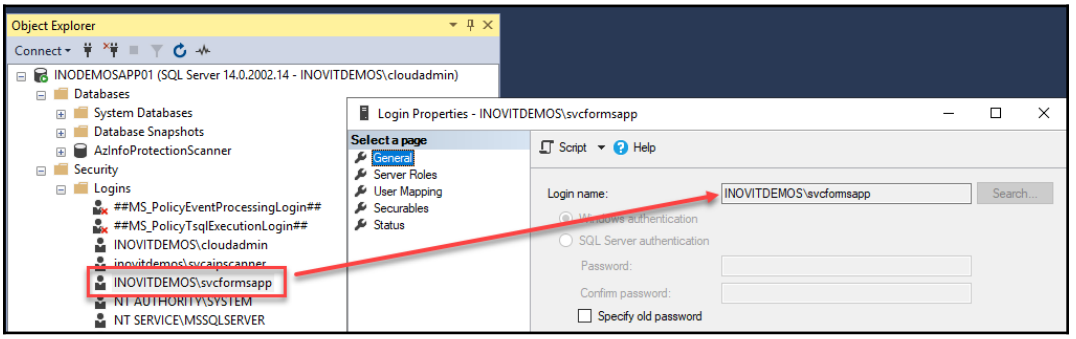
```
New-ADUser -Name "svcformsapp" -SamAccountName svcformsapp -
UserPrincipalName svcformsapp@inovitdemos.ch -path
"OU=Users,OU=AAD,OU=Managed Service
Objects,DC=inovitdemos,DC=ch" -AccountPassword (ConvertTo-
SecureString "MIA@me1976ch" -AsPlainText -Force) -Description
"Forms App Pool Account" -Enabled $True
```

15. Connect to your **SQL Server** over in the SQL Management Studio on your **YD1APP01:**



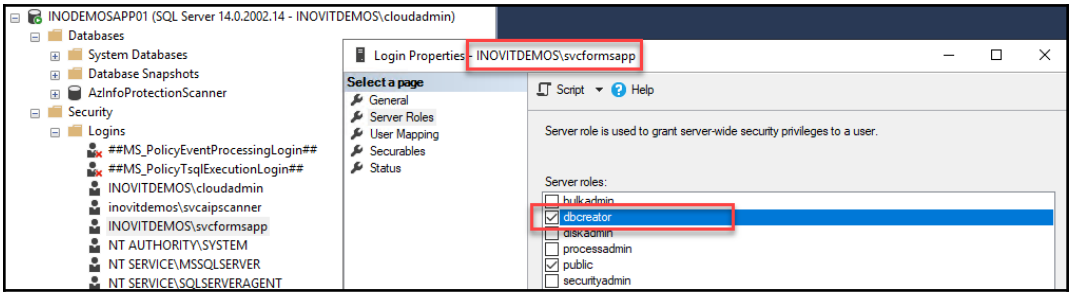
SQL Connection with the SQL Management Studio

16. Create a login for your service account under **Security** | **Logins**:



Configuring the login for the app service account

17. Assign the **dbcreator** server role to the user:

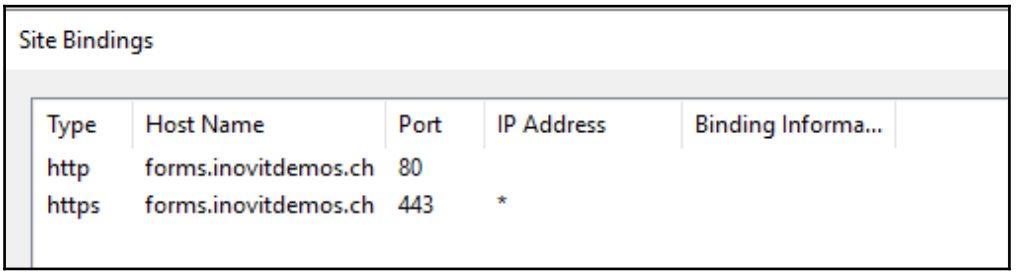


Assigning the dbcreator role to the service account

18. Next, we need to create the website on the server:

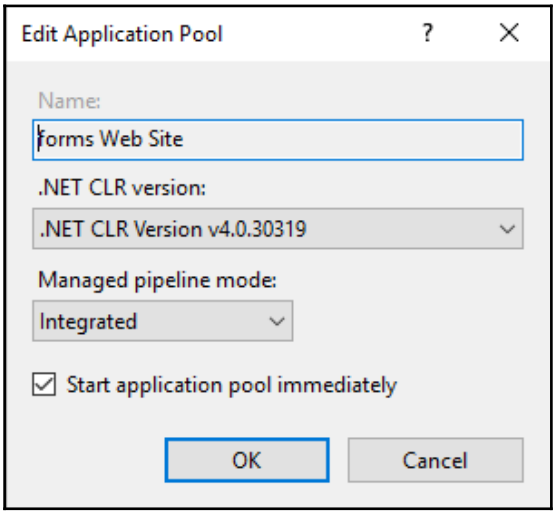
```
New-Item C:\inetpub\formsroot -type Directory
Import-Module Webadministration
cd IIS:
New-Item 'IIS:\Sites\forms Web Site' -bindings
@{protocol="http";bindingInformation=":80:forms.inovitdemos.ch
"} -physicalPath 'c:\inetpub\formsroot'
```

19. As with the other on-premises demo applications, we will create a HTTPS binding and assign our SSL certificate:



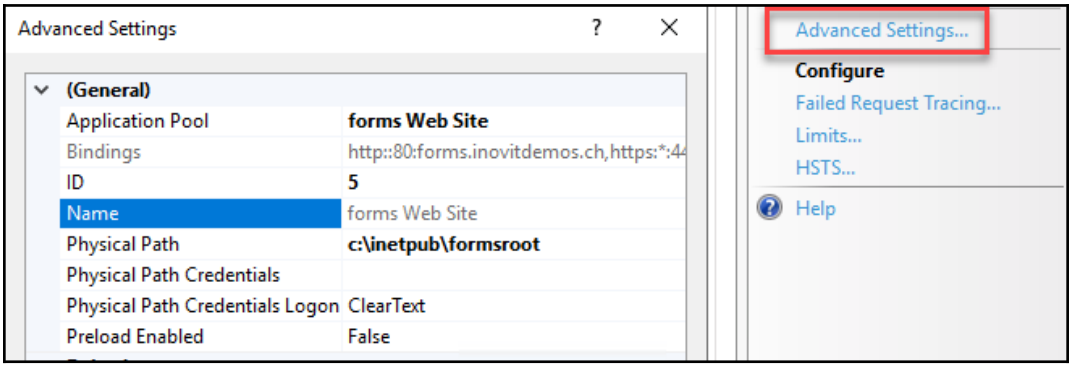
Configuring the IIS Bindings for the application

20. We also need to create a new app pool for the website and use the service account to run the app:



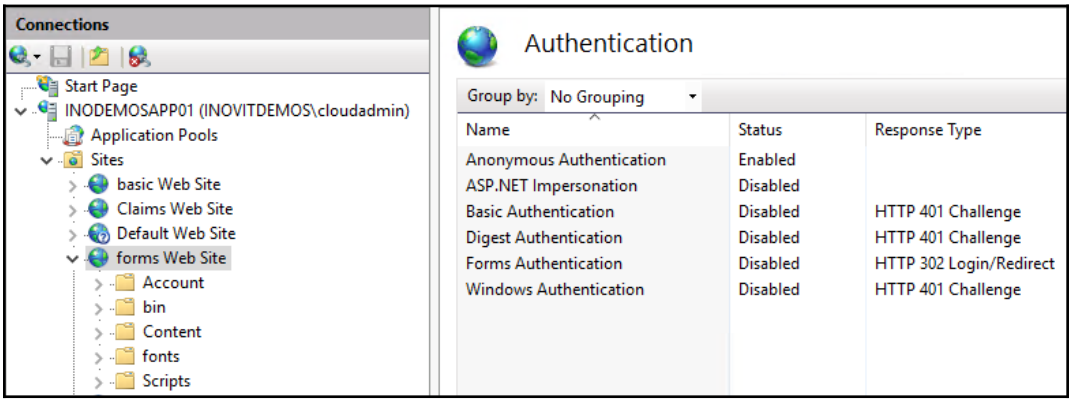
Creation of the Application Pool

21. Next, we will assign the newly created application pool to our website under **Advanced settings**:



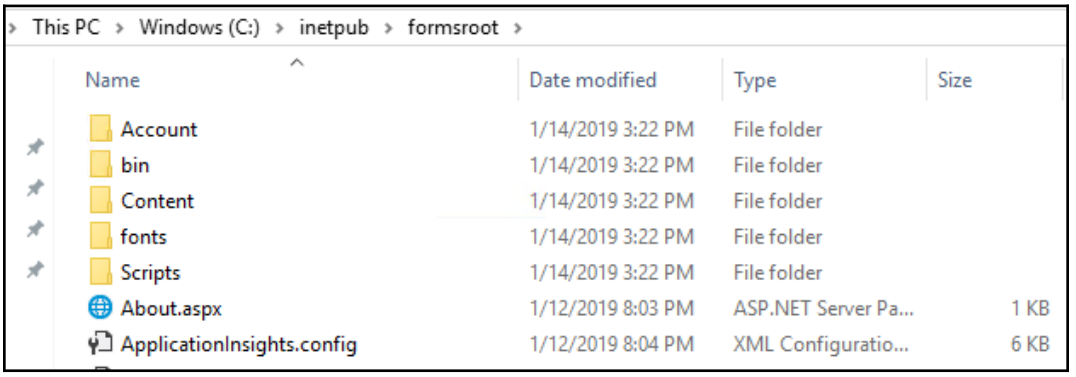
Assigning the newly created app pool

22. The authentication for the page should be configured as shown in the following screenshot:



IIS Authentication configuration

23. The next step is that we need to copy the content of the `formsapp` folder from the code package to `C:\inetpub\formsroot`:

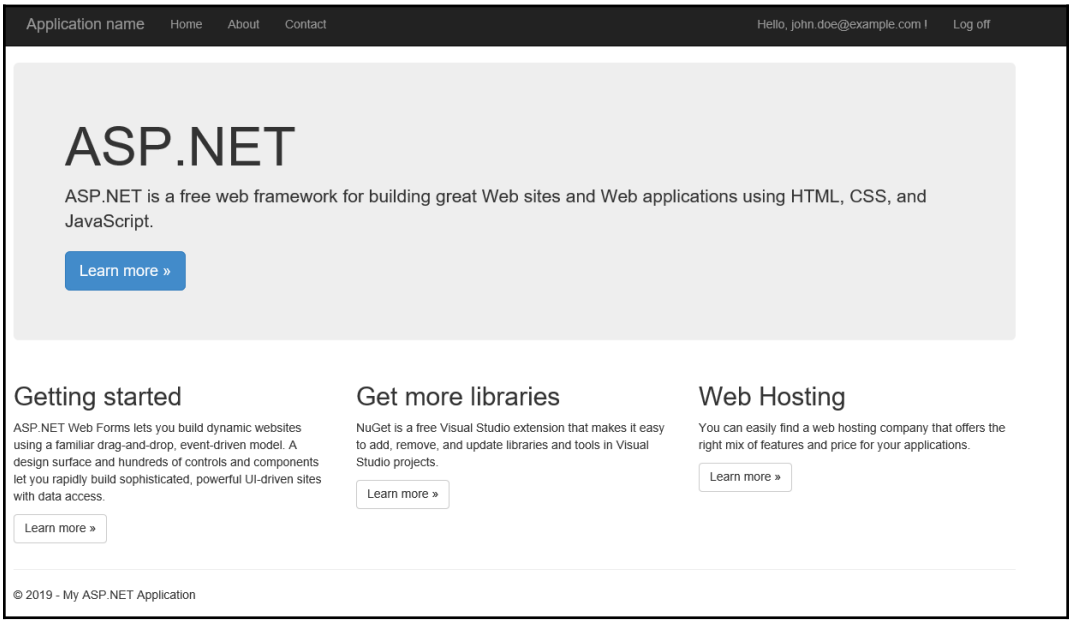


Application Code

24. Before you can run the app, you need to configure the `web.config` file to address the **SQL Server** instance:

```
<add name="DefaultConnection" connectionString="Data
Source=YD1APP01;Initial
Catalog=FormsBasedAuthentication;Integrated Security=True"
providerName="System.Data.SqlClient" />
```

25. Now, you can test your application by registering a user and a successful login:



Successful test of the app

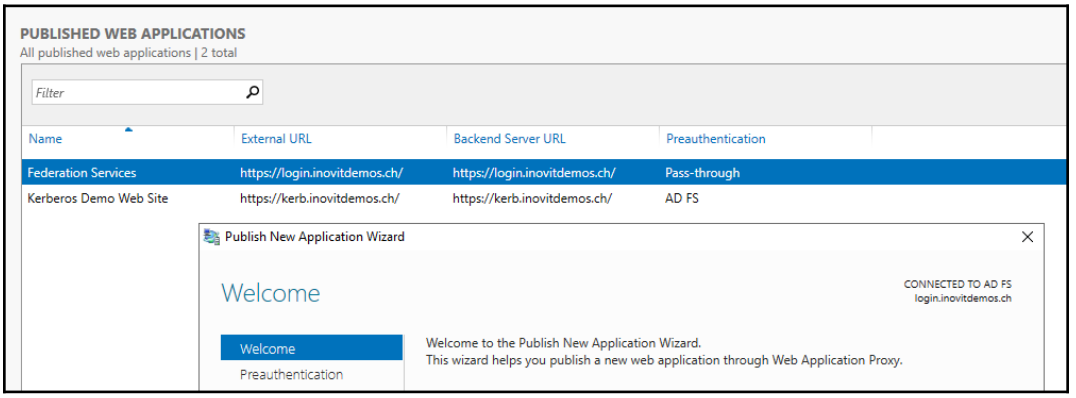
26. Start with your publishing scenario and log in to this website—good luck!

In the next example, we will use the **Windows Server Web Application Proxy** to publish a Basic authentication-based application to external users:

1. First, you need to log in to your **YD1ADS01** to configure the relying party for the Basic demo application.
2. Navigate to Relying Party Trusts and Add Relying Party Trust.
3. Choose the non-claims-aware option.
4. Fill in the following Display Name: `Basic Demo Web Site`.
5. Use `https://basic.inovitdemos.ch` (replace it with your domain) as Relying Party Trust Identifier and click **add**.

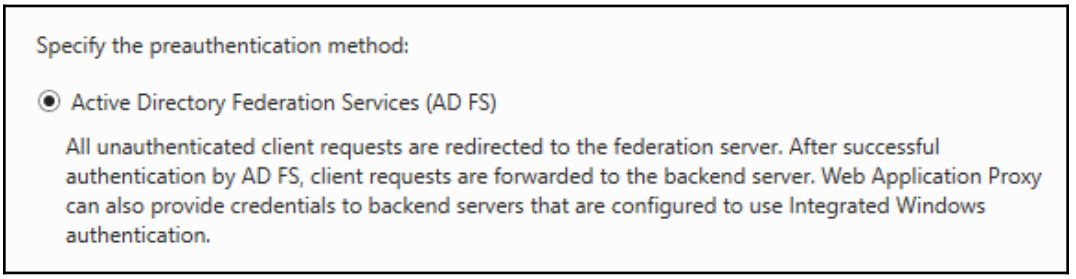
Don't forget to configure the FQDN in your public DNS.

6. Click Next until the wizard is finished.
7. Log in to your **YD1URA01** server and open the Remote Access Management console.
8. Click on **Publish** in the right-hand Tasks pane:



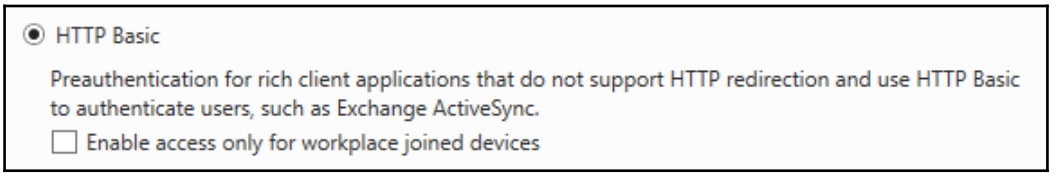
Publishing wizard on the web application proxy

9. Specify the **ADFS preauthentication method**:



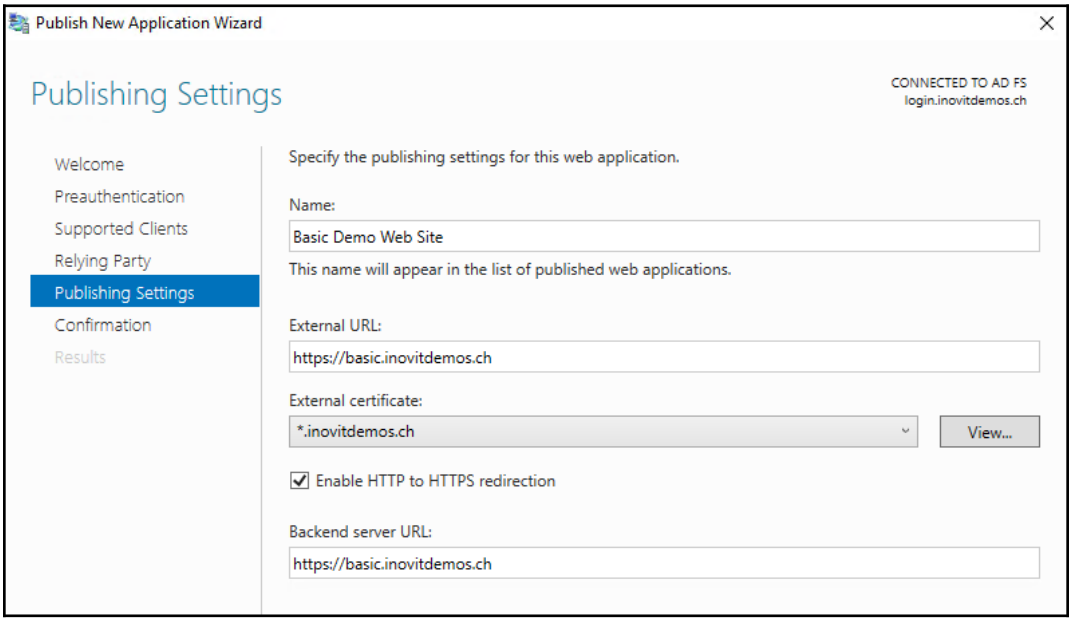
Using the ADFS preauthentication method

10. Choose **HTTP Basic** as the type of **preauthentication**:



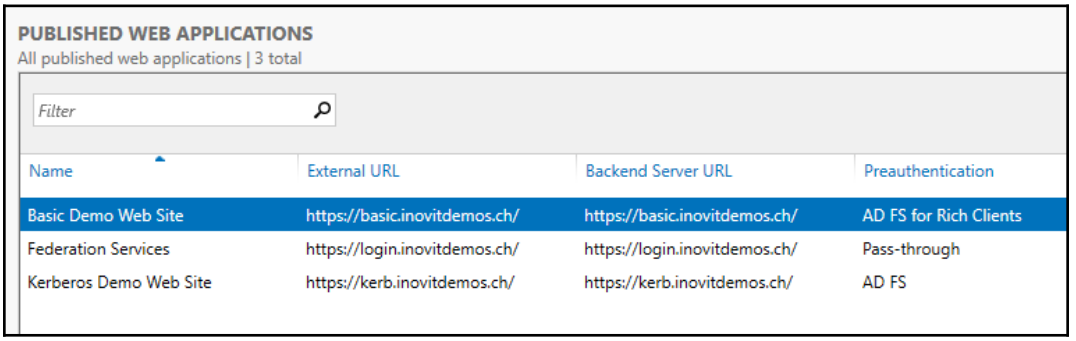
Using HTTP Basic

11. Choose **Basic Demo Web Site** relying party.
12. Fill in the following publishing values:



Setting the application properties

13. Click **Next**, **Publish**, and **Close**:



Final publishing settings

14. Now, you can test your freshly published Basic authentication app.

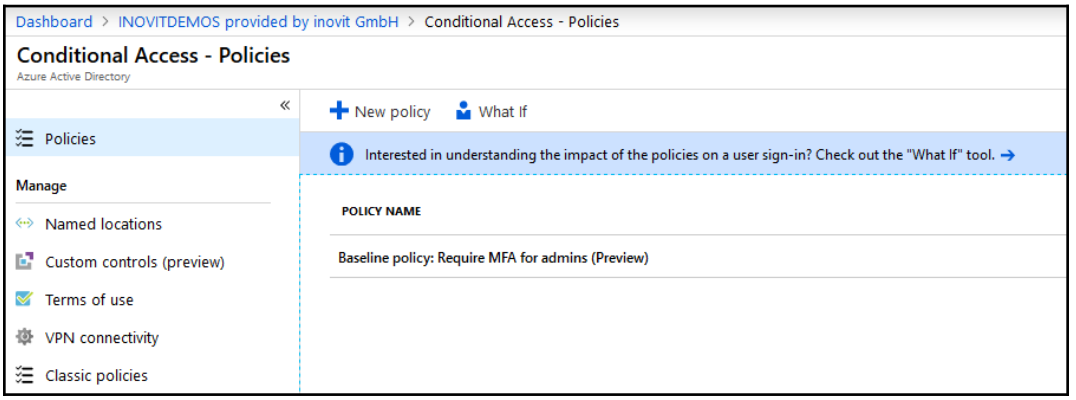
You will find more practical scenarios for several services, such as the Exchange and Remote Desktop services, at the following sources: `https://docs.microsoft.com/en-us/windows-server/remote/remote-access/web-application-proxy/publishing-applications-using-ad-fs-preauthentication` and `https://docs.microsoft.com/en-us/windows-server/remote/remote-access/web-application-proxy/publishing-applications-using-ad-fs-preauthentication`.

In the next section, we will include the first conditional access options.

Using conditional access

In our first conditional access scenario, we will use the Azure AD functionality to secure **Salesforce** access with Azure MFA:

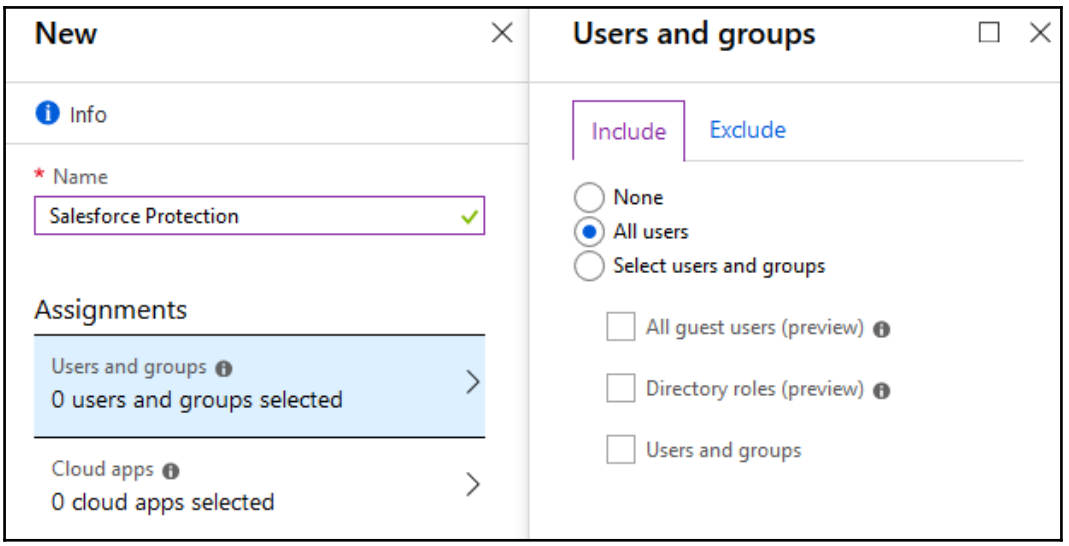
1. Navigate to `https://portal.azure.com` and the Azure AD pane | **Conditional Access**.
2. Click **New policy**:



Creating a Conditional Access policy

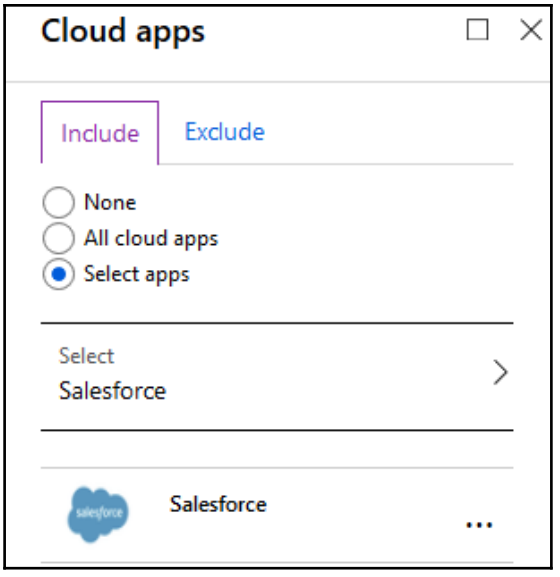
3. Call the new policy `Salesforce Protection`.

4. Under assignments, go to **Include** | **All users**:



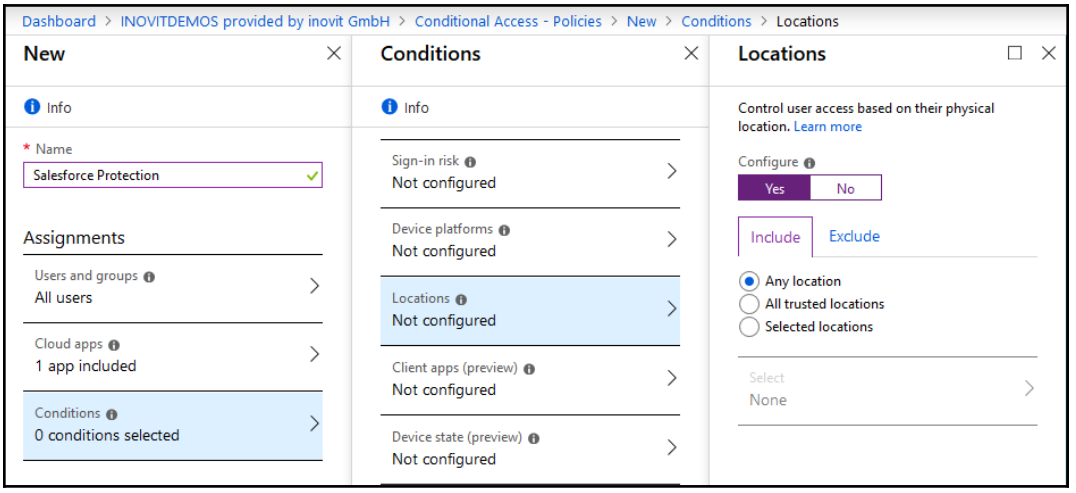
User assignment options

5. Under **Cloud apps** | **Select apps**, choose **Salesforce**:



Selecting the Salesforce app

6. Under **Conditions** | choose **Locations** | **Yes** and Any location:

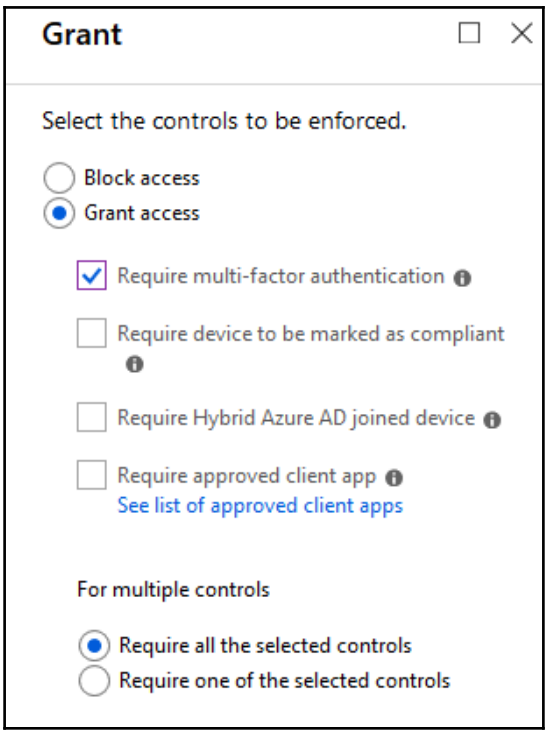


Choosing the location attributes

As you can see, you have many conditions that can be set when you want to fulfill security requirements in the case of additional authentication or access control mechanisms. You can find more information at the following source: `https://docs.microsoft.com/en-us/azure/active-directory/conditional-access/`.

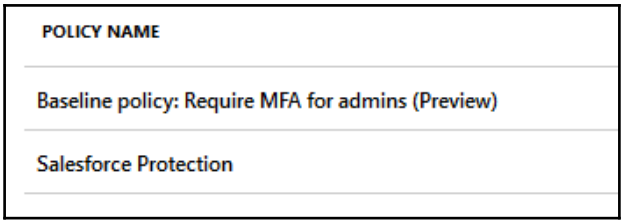
7. Under Access controls, go to **Grant**.

8. Choose **Grant access** | **Require multi-factor authentication**:



Using MFA for granting the access

9. Enable the policy and click **Create**:

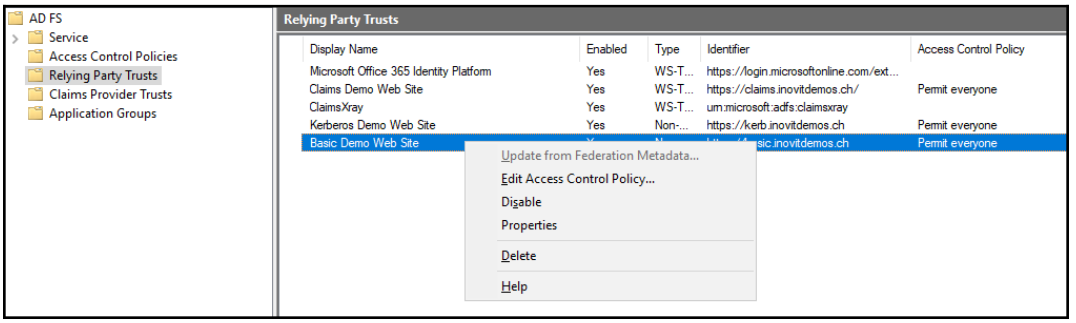


Enabling the policy

10. Now, you can test it with a **Salesforce** assigned user and Azure MFA will be required.

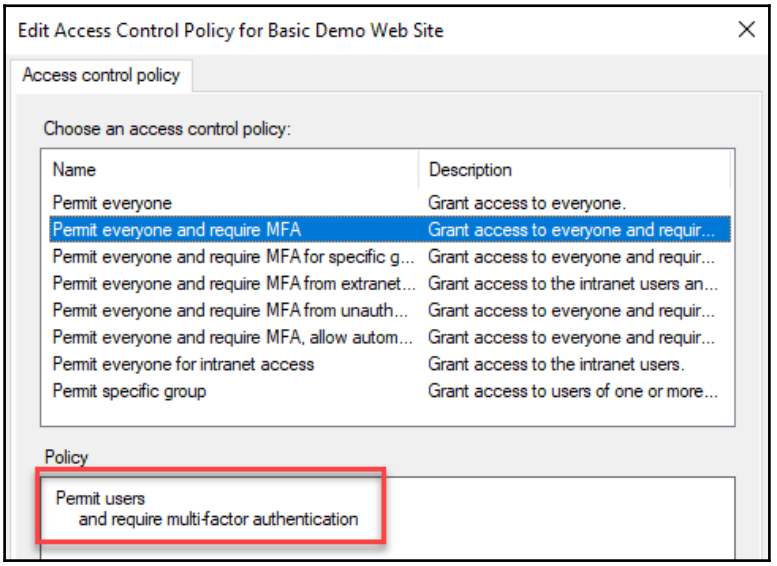
In the second example, we will follow the same scenario with our on-premises ADFS infrastructure:

1. Log in to your ADFS server (**YD1ADS01**) and open the ADFS management console.
2. We will enable **Azure MFA** for our **Kerberos Demo Web Site.**
3. Right-click on the app and choose **Edit Access Control Policy**:



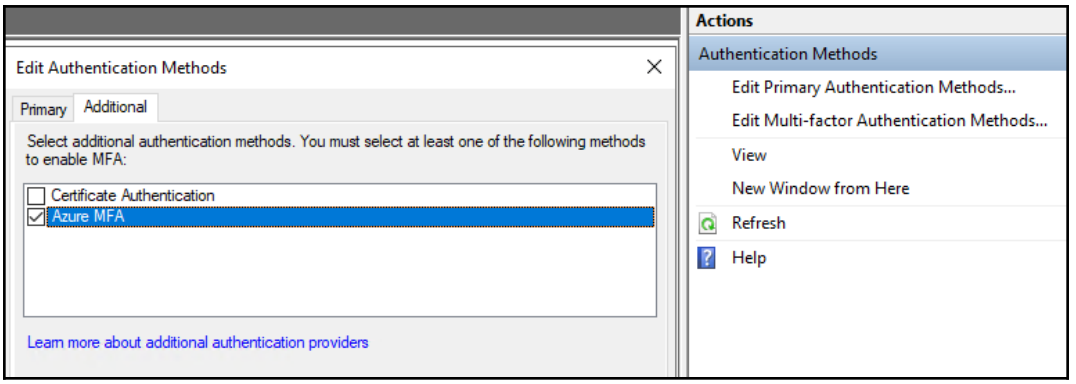
Configuring Access Control Policies for the app

4. Next, choose **Permit everyone and require MFA**:



Enabling the MFA option

5. Apply the settings and navigate to **Authentication Methods**.
6. Choose **Edit Multi-Factor Authentication Methods**.
7. Go to Additional and choose **Azure MFA**:



Choosing the Azure MFA method

8. Apply the setting and you are now ready to test the application.

You need to use a registered Azure MFA user to access the application. In the code package for the chapter, you will find additional use cases, including the customization required to modify the login web page to interact with the user.

We will include more conditional access use cases and more information about the Azure MFA service in the following chapters:

- Exploring Azure AD identity services
- Developing single-and multi-tenant applications
- Configuring Azure information protection solutions

This is to show its relevance to specific requirements.

Summary

In this chapter, you have learned to configure different authentication scenarios using Azure AD and ADFS capabilities. There are many possible combinations, and we can't provide all of them in one chapter. We gave you an introduction and some of our practical tips to get started and guide you in your upcoming journey through this technology. Don't worry; we jump more and more into some different scenarios in the next chapters of this book to give you as much help as possible. Also, look at the code package for the book; you will find additional practical examples from our projects.

In the next chapter, we will explore more Azure AD identity services, such as Azure AD B2B and B2C.

9
Deploying Additional Applications on Azure AD

Providing the correct authentication to your application is essential, particularly if you want to allow authentication only to users from your **Azure Active Directory** (**Azure AD**), or from any other Azure AD. Azure AD provides two concepts to give your users the required authentication.

In this chapter, we'll introduce you to the idea of single-tenant and multi-tenant applications and what's different between them. Furthermore, we'll discuss roles and claims in a single-tenant app that we move to a multi-tenant app, and you can test the transition between the two models. We'll also deploy a multi-tenant app that uses OpenID Connect. In all the different labs, you'll learn what you need from an application vendor to integrate the application into your environment. You'll be able to ask the right questions and take your deployment in the right direction.

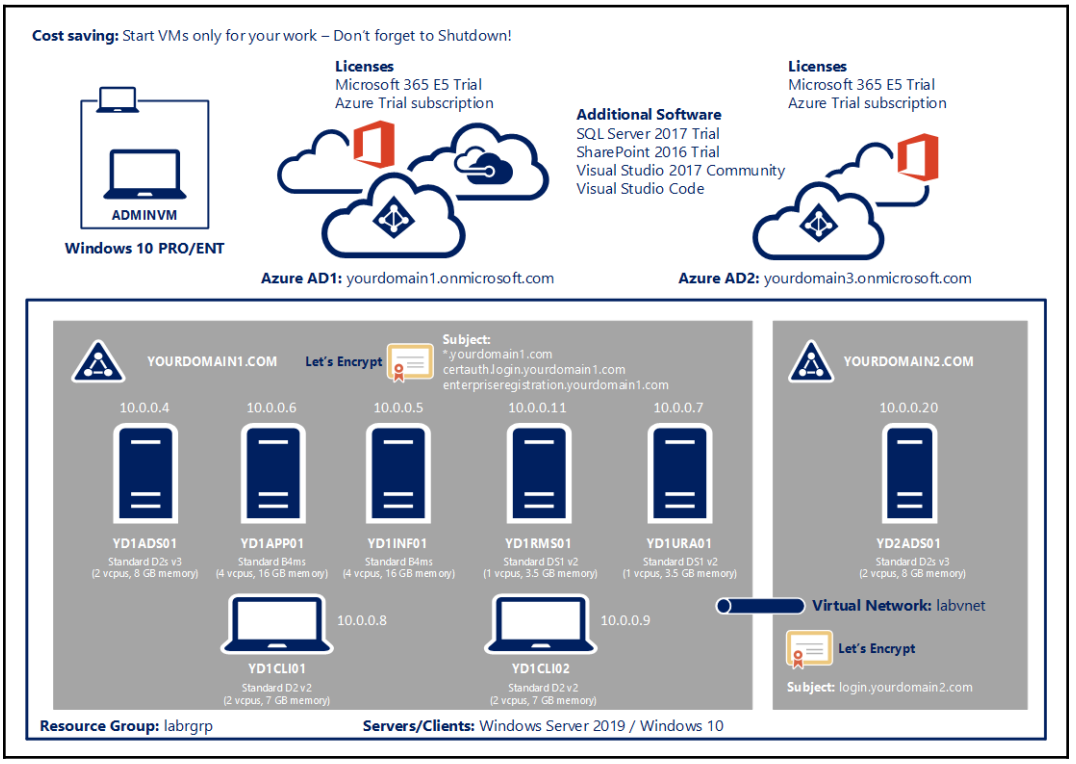
We'll divide the chapter into the following sections:

- Preparing your lab environment
- What defines a single- and multi-tenant application
- Deploying a single-tenant application, including roles and claims
- Moving the single-tenant app to a multi-tenant scenario
- Deploying a multi-tenant app with OpenID Connect

As usual, we'll do a lot of practical tasks. Let's organize the prerequisites to handle the labs in your demo environment.

Preparing your lab environment

In this chapter, you can use any machine that has Visual Studio installed. In our case, we installed it on the **YD1APP01** server. This should be your perfect development environment. You will use **Azure AD1: yourdomain1.onmicrosoft.com** during this lab, and one test guest user from another tenant and a test user from any other Azure AD to test multi-tenant applications. Furthermore, you will need to use the code examples from the code package to use it in your Visual Studio to configure and run the applications:



Lab environment overview

After preparing our lab environment, we can directly jump into the first section of the chapter, which describes single- and multi-tenant applications.

What defines single- and multi-tenant applications

On the face of it, there's a simple explanation of what a single-tenant or a multi-tenant application is. We can say that single-tenant apps are only available in the tenant they were registered. On the other hand, multi-tenant apps are available to users in both your home tenant and in other tenants:

- Single-tenant applications are primarily used if you want to isolate your application from everyone and provide access only to your internal users
- Multi-tenant applications are used when you want to provide the app to your internal staff, guest users, and Microsoft personal account users in a collaboration scenario, for example

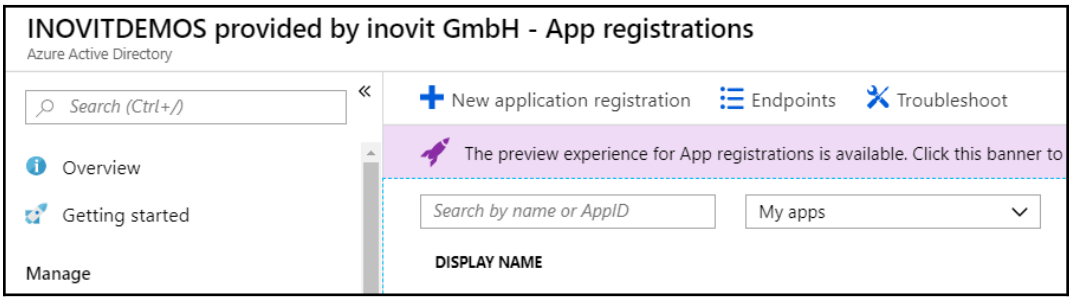
We highly recommend reviewing `https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-convert-app-to-be-multi-tenant` to find the differences between the endpoints that are used in both scenarios.

Deploying a single-tenant application including roles and claims

In this section, we'll deploy Microsoft's sample, `WebApp-RoleClaims-DotNet.sln`, as a single-tenant configuration. We start with this setting and move the application to a multi-tenant application in the next part. The tracker app provides the following application roles, which we can use to test the role/claims topic. First, we can use the admin role to perform all actions. With the writer role, you're empowered to create tasks in the application. To change the status of a task, you can assign the approver role. To view the tasks and their associated states, we can map the observer role.

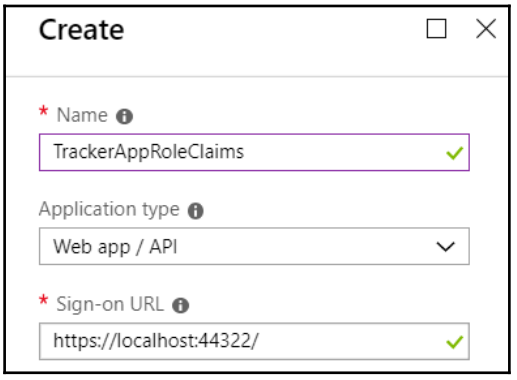
With the following steps, we'll configure Azure AD for our application:

1. Open the Azure portal: `https://portal.azure.com`.
2. Navigate to the Azure AD blade.
3. Click **App registrations**.
4. Create **+New application registration**:



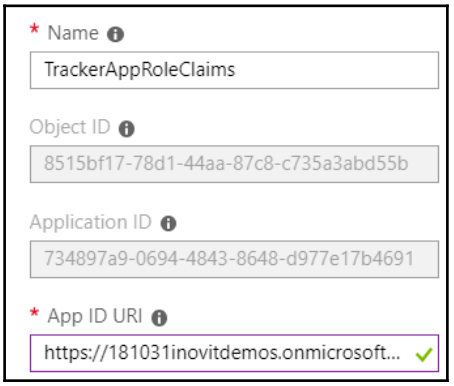
New app registration

5. Provide a name.
6. Add the **Sign-on URL**, `https://localhost:44322/`:



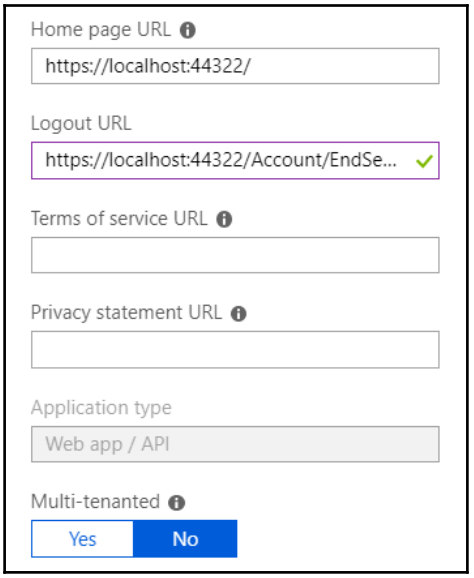
App settings dialog

7. Provide the **App ID URI** in the format `https://181031inovitdemos.onmicrosoft.com/trackerapp`:



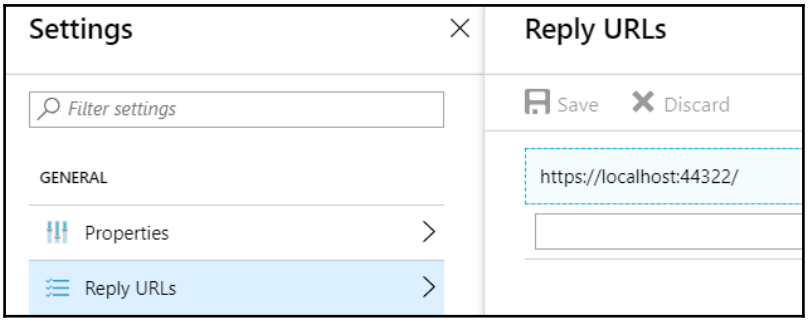
Additional app settings dialog

8. Provide the **Logout URL**, `https://localhost:44322/Account/EndSession`.
9. Leave the other settings as the defaults:



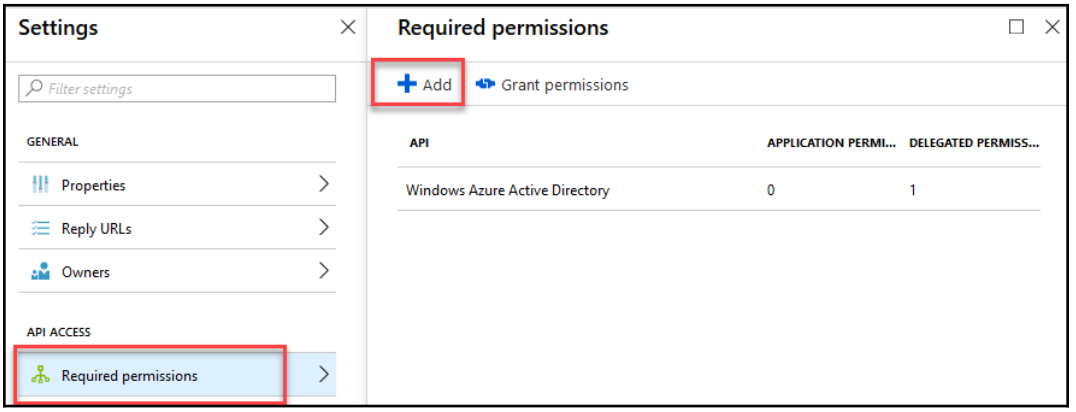
Logout URL configuration

10. Check the **Reply URLs**; they need to look like the following figure:



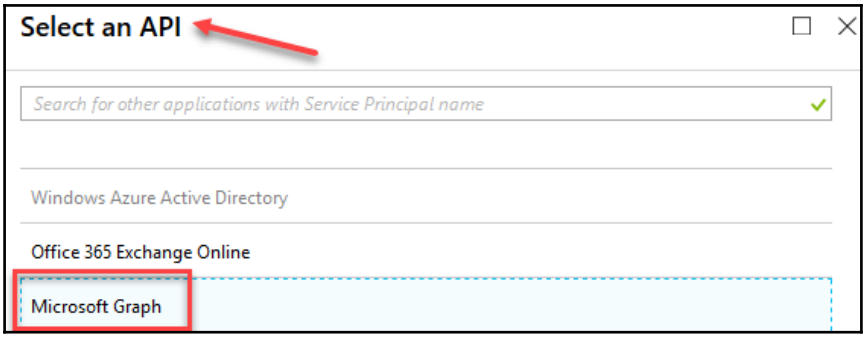
Reply URL configuration

11. Under **Required permissions,** click **Add**:



Permission configuration

12. Select the **Microsoft Graph** API:

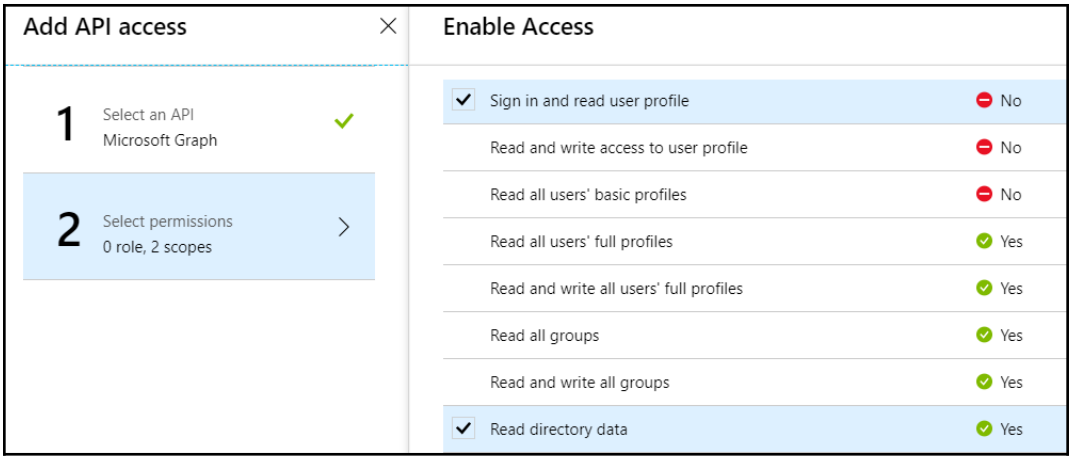


Choosing the Microsoft Graph API

13. Select the following permissions:
    - Sign in and read user profile
    - Read directory data

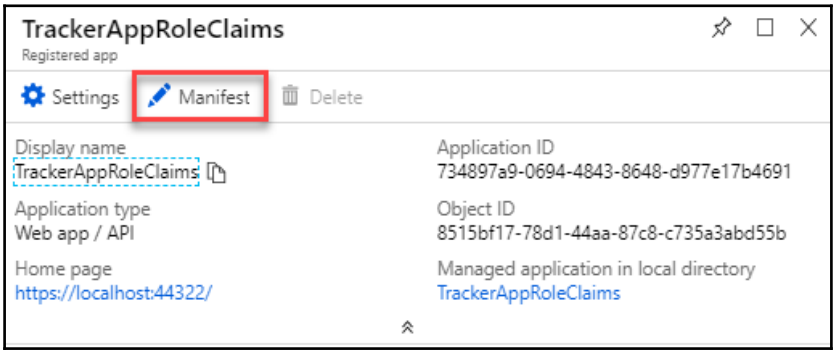
You can use the Microsoft Graph permission reference `https://docs.microsoft.com/en-us/graph/permissions-reference` to learn more about this topic.

14. The selected permissions should look like the following screenshot:



Assigning the two needed scopes

15. Save and click **Grant Permissions**.
16. Open the **Manifest** of the application:



Configuring the Manifest

17. Configure the roles of the following figure in the `appRoles` section with the configuration code of the code package.

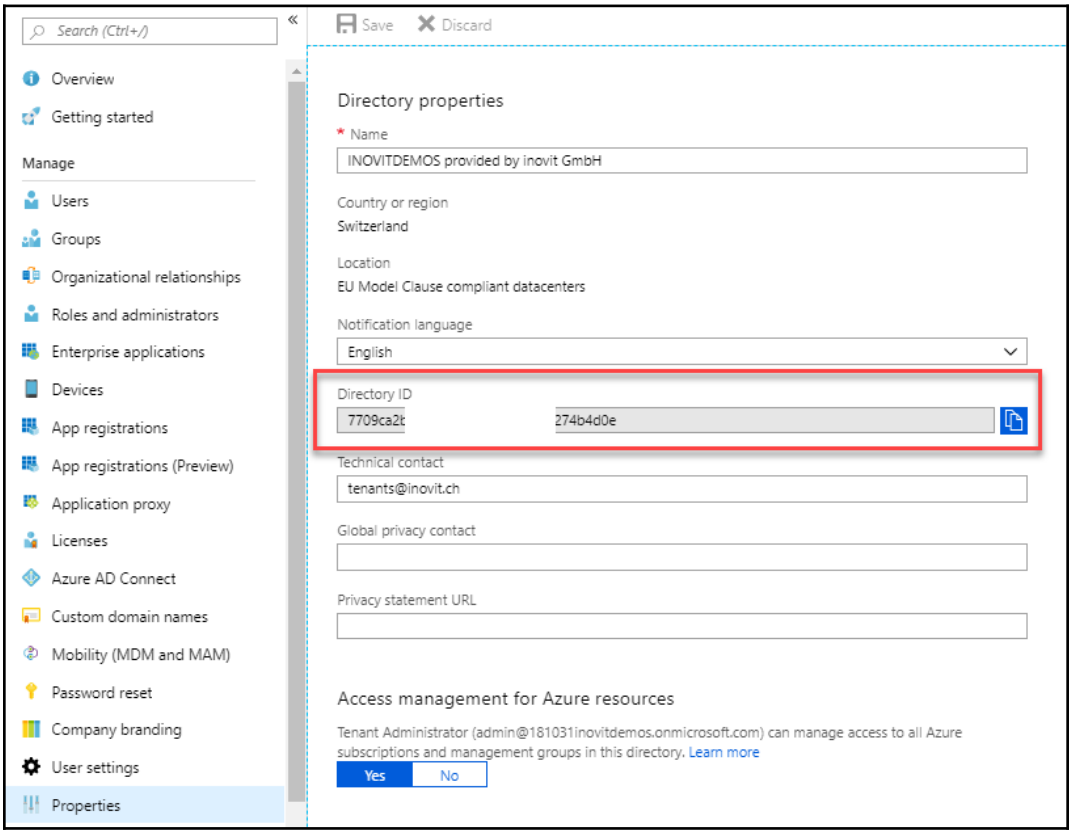
18. The result in your manifest should look like this:

```
  "appRoles": [
{
    "allowedMemberTypes": [
      "User"
    ],
    "displayName": "Writer",
    "id": "d1c2ade8-98f8-45fd-aa4a-6d06b947c66f",
    "isEnabled": true,
    "description": "Writers Have the ability to create tasks.",
    "value": "Writer"
  },
  {
    "allowedMemberTypes": [
      "User"
    ],
    "displayName": "Observer",
    "id": "fcac0bdb-e45d-4cfc-9733-fbea156da358",
    "isEnabled": true,
    "description": "Observers only have the ability to view tasks and their statuses.",
    "value": "Observer"
  },
  {
    "allowedMemberTypes": [
      "User"
    ],
    "displayName": "Approver",
    "id": "fc803414-3c61-4ebc-a5e5-cd1675c14bbb",
    "isEnabled": true,
    "description": "Approvers have the ability to change the status of tasks.",
    "value": "Approver"
  },
  {
    "allowedMemberTypes": [
      "User"
    ],
    "displayName": "Admin",
    "id": "81e10148-16a8-432a-b86d-ef620c3e48ef",
    "isEnabled": true,
    "description": "Admins can manage roles and perform all task actions.",
    "value": "Admin"
  }
  ],
```

Adding the app roles

19. Save the manifest.
20. Copy the app ID to Notepad.

21. Navigate to your Azure AD tenant's **Properties**.
22. Copy the **Directory ID** to Notepad:



Getting the Directory ID

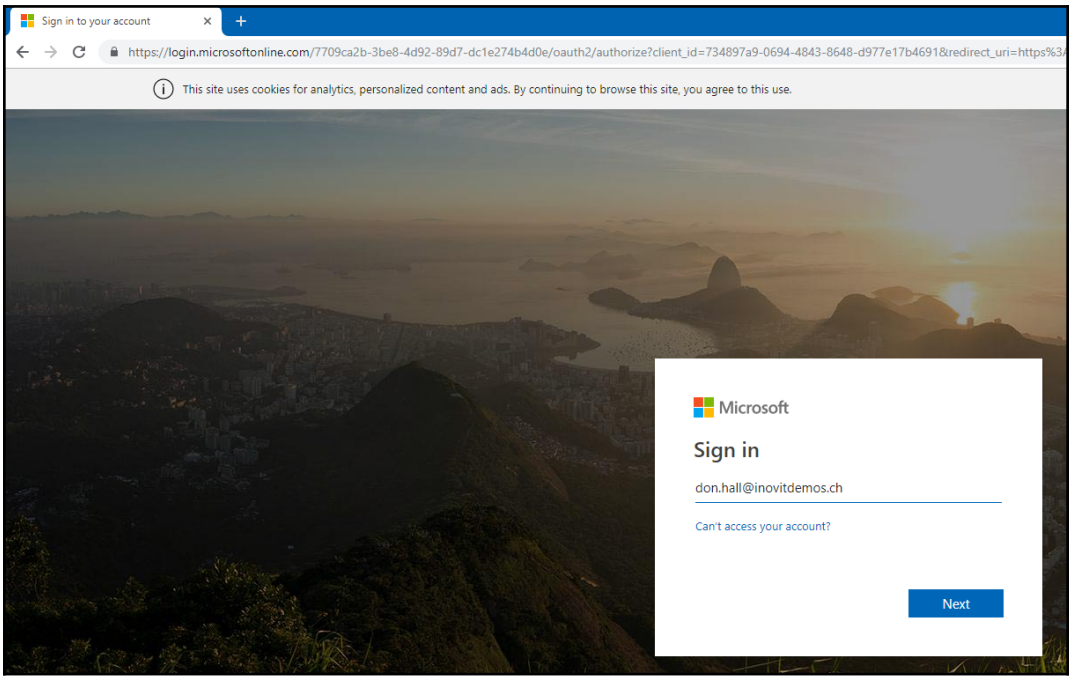
23. Open your Visual Studio and the solution file from the code package named `WebApp-RoleClaims-DotNet.sln`.
24. Open the `web.config` file of the solution and do the following:
    - Enter the app ID on the `ida:ClientId` value
    - Enter the **Directory ID** on the `ida:TenantId` value

25. The following result is expected:

```
<appSettings>
  <add key="webpages:Version" value="3.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  <add key="ida:AADInstance" value="https://login.microsoftonline.com/{0}" />
  <add key="ida:ClientId" value="734897                    e17b4691" />
  <add key="ida:TenantId" value="7709ca                    274b4d0e" />
  <add key="ida:PostLogoutRedirectUri" value="https://localhost:44322/" />
```

Modifying the web.config file

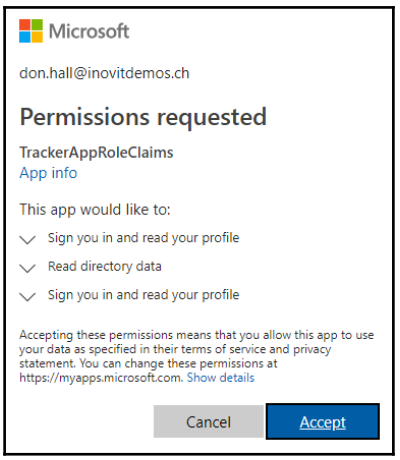
26. Press *F5* in Visual Studio to build and run the application.
27. Sign in as Don Hall:



Testing sign-in functionality

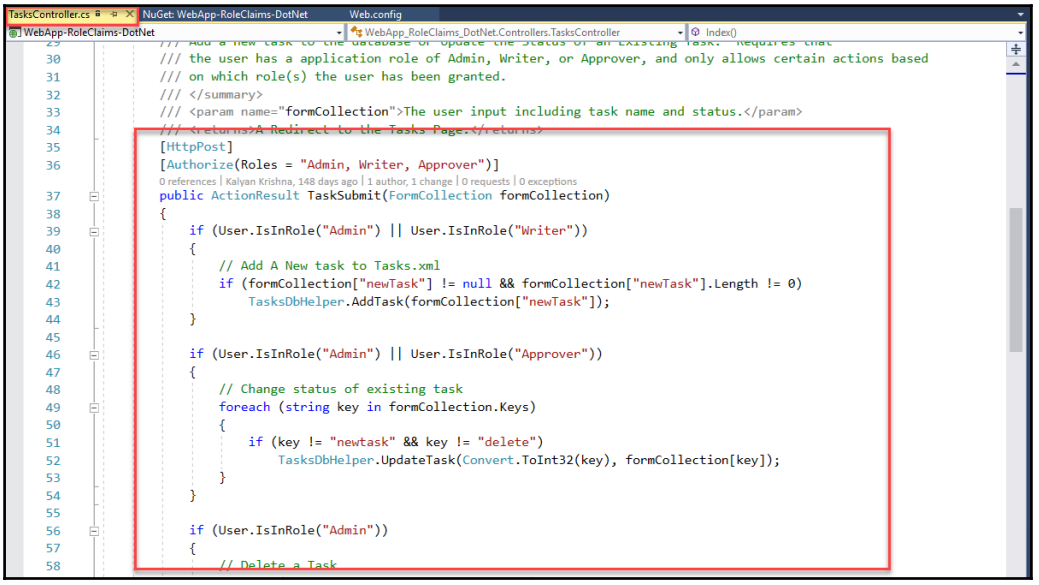
28. Consent to application.

29. Click **Accept**:



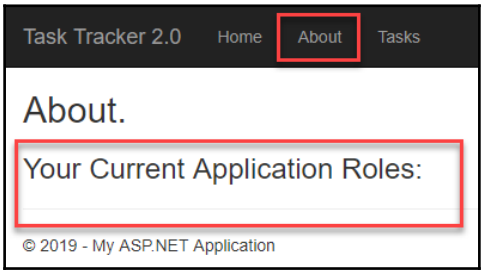
OAuth consent

30. Go back to Visual Studio and review the role implementation in the `TaskController.cs` file:



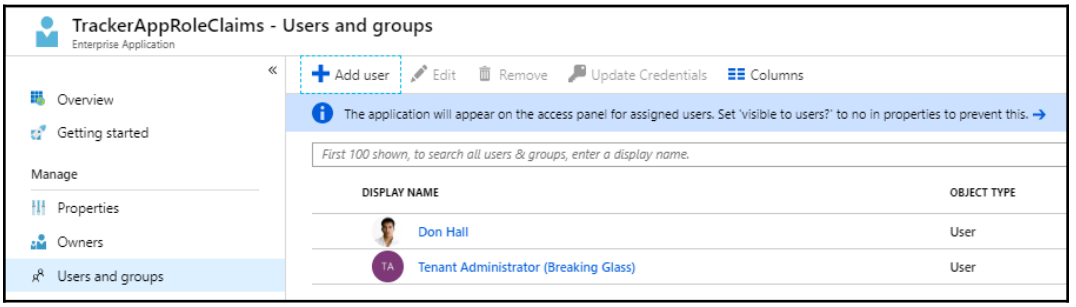
Reviewing the role implementation in the code

31. Don Hall will be signed in.
32. Click **About** to view the actual roles; no roles are assigned:



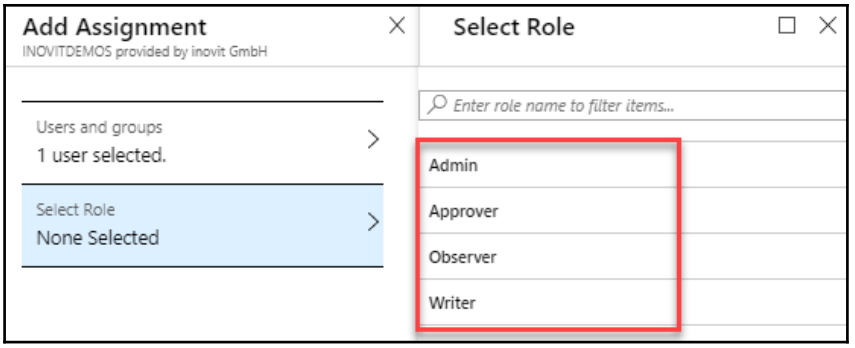
Viewing the actual app roles

33. Go back to your application and assign roles to several test users.
34. Click **Add user**:



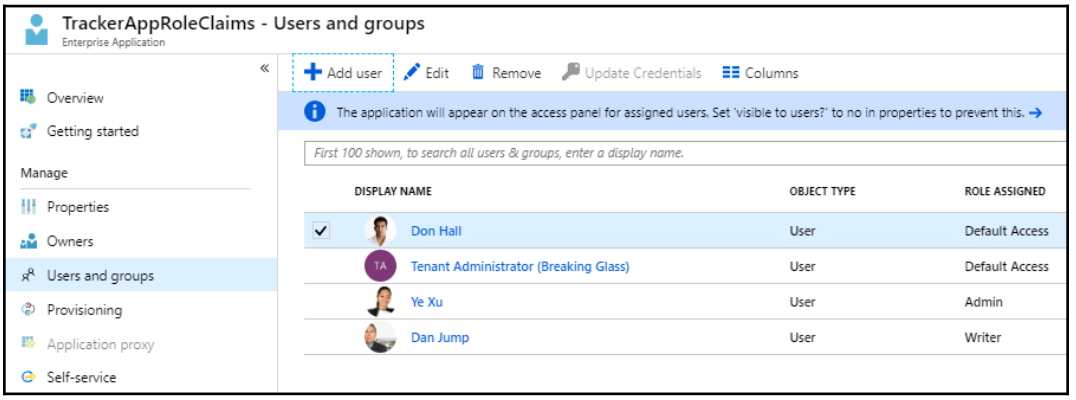
Assigning users and groups

35. Assign the role you want to test:



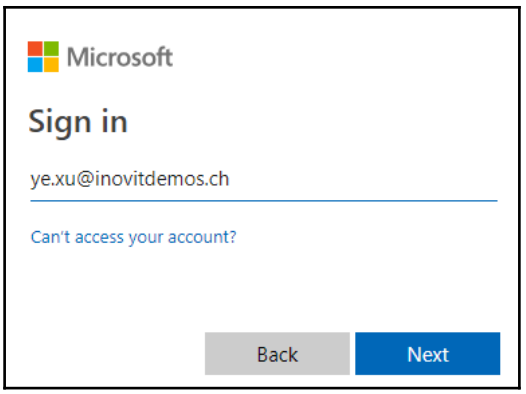
Select the appropriate role

36. In my case, I used two accounts, as you can see in the following screenshot:



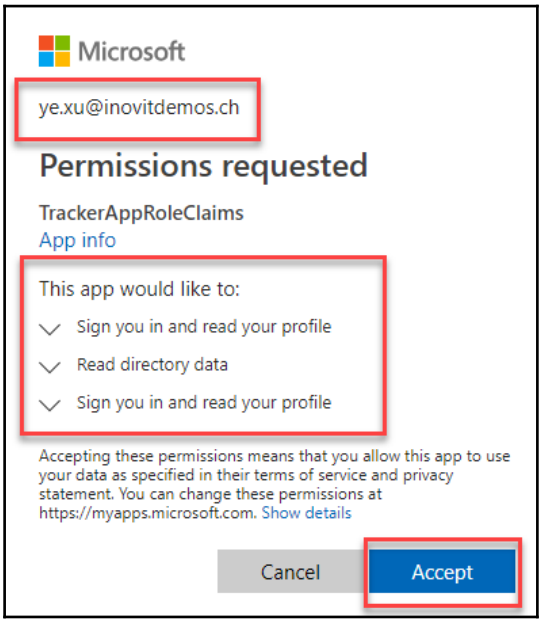
Role assignment result

37. Try to sign in with your users; I started with Ye Xu (Admin role assigned):



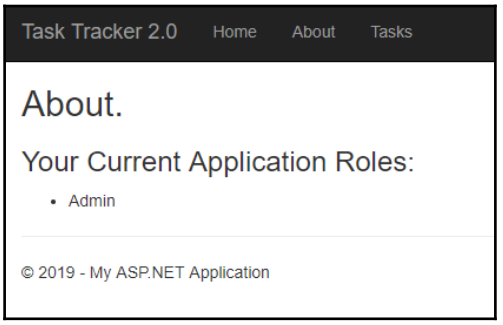
Testing sign-in functionality

38. Accept the user consent:



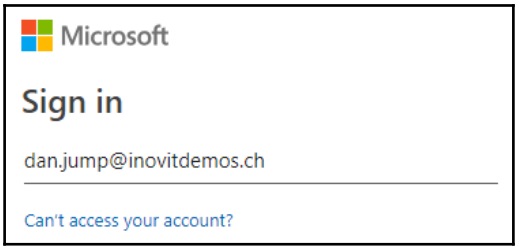
Checking and approving the OAuth consent

39. Click **About** to view the assigned role. We can see Ye Xu has the **Admin** role assigned:



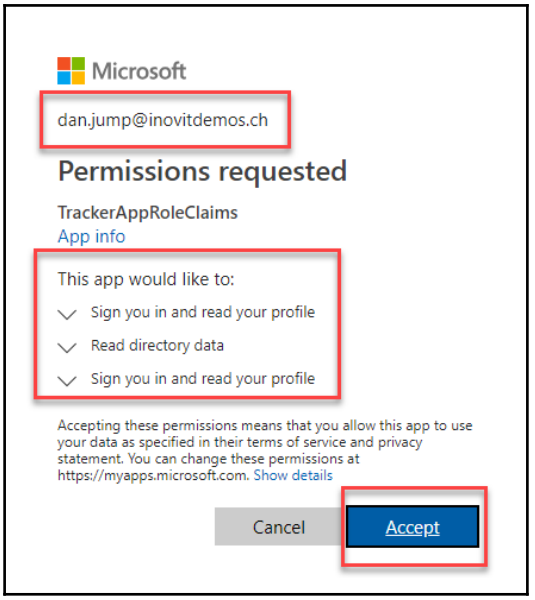
Check your current role

40. Try the user with a different role, in my case Dan Jump (**Writer** role assigned):



Testing with the second test user

41. **Accept** the consent:

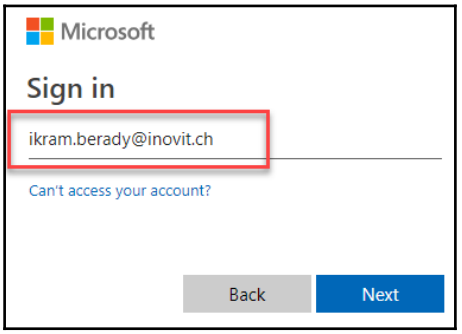


Checking and approving the OAuth consent

42. You will see the `Writer` role is correctly assigned:

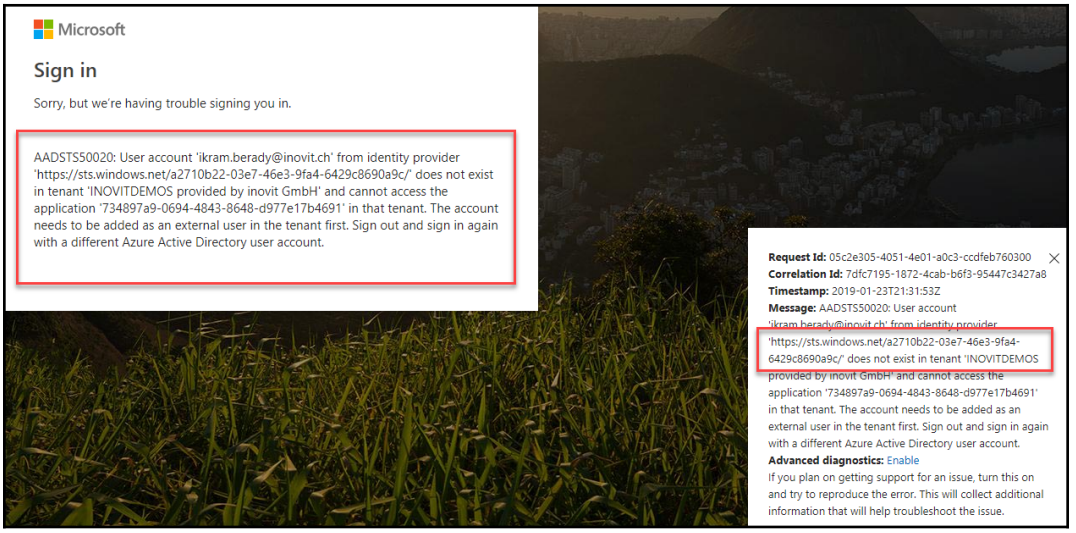
View your current role

43. Try to sign in with a user that's not part of the organization (one from another Azure AD):



Test with a guest user account

44. Remember, we're using a single-tenant configuration at the moment. The user gets an error message and can't sign in:



Review the error messages

Our demo app is running and working with role assignments for users of our own Azure AD tenant. Other users from other Azure AD tenants can't access the application right now.

We highly recommend working through the following guidance to customize claims in Azure AD for other protocols, such as SAML:

- Enterprise role app management: `https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-enterprise-app-role-management`
- SAML claims customization: `https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-saml-claims-customization`
- Claims mapping: `https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-claims-mapping`
- Configuring optional claims: `https://docs.microsoft.com/en-us/azure/active-directory/develop/active-directory-optional-claims`

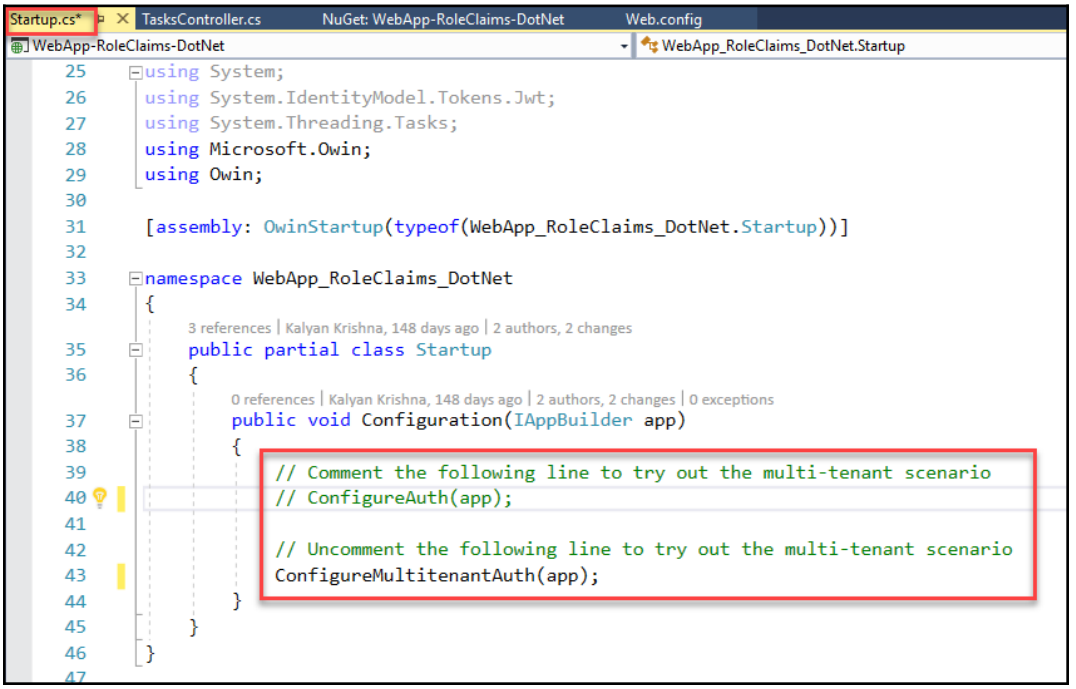
Next, we'll move the app to a multi-tenant configuration.

Moving the single-tenant app to a multi-tenant scenario

In this section, we'll reconfigure the application to work as a multi-tenant application that you can use from other Azure AD tenants or a Microsoft personal account. With the following configuration we migrate the application:

1. Open the `Startup.cs` file in your solution:
    - Comment the `ConfigureAuth(app)` line

- Uncomment the `ConfigureMultitenantAuth(app)` line:

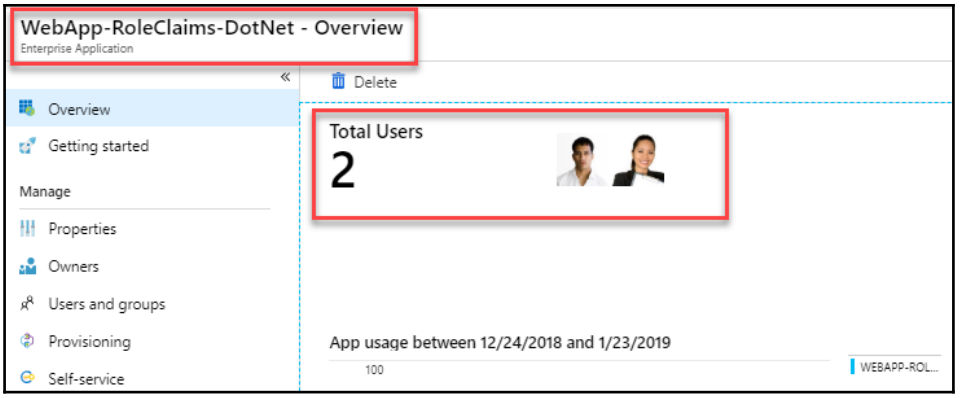


Modifying the code for multi-tenant usage

2. Change the `ida:TenantId` value to our Azure AD domain name:

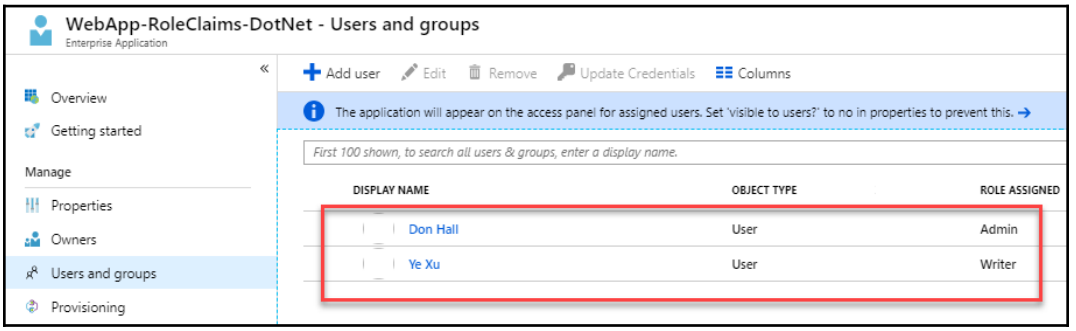
Changing the tenant ID

3. Press *F5* in Visual Studio to build and run the app.
4. A new app will be available in Azure AD, but the users don't have roles assigned:



Check the role assignment for the newly created app

5. Reassign the roles to the users:



Reassign roles

6. Test logging in with your users and check the application.
7. Test logging in with a user from another Azure AD or a Microsoft personal account.
8. You should be able to log in:

Check the login with the guest user account

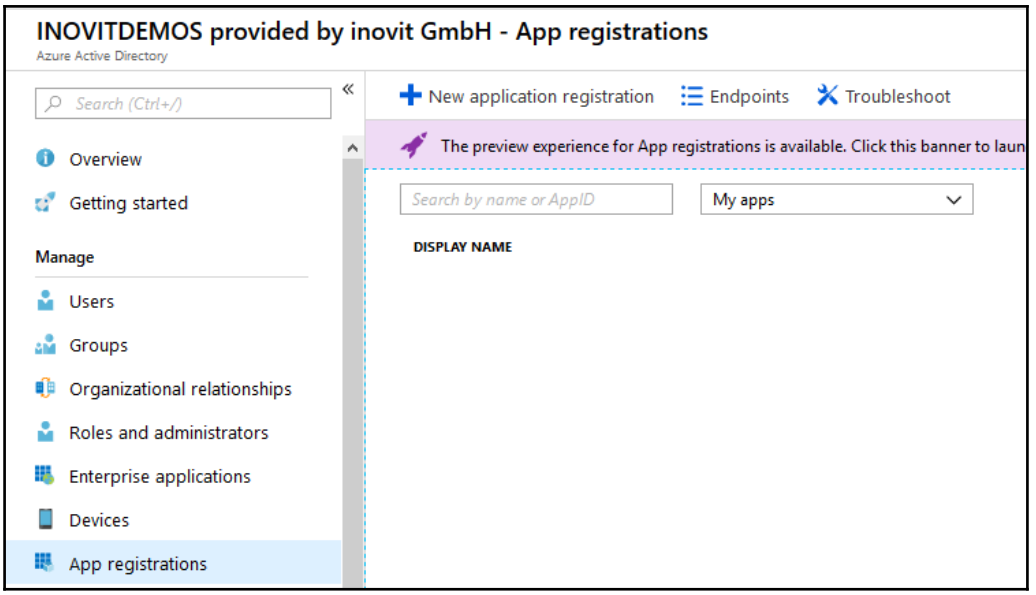
You can find more information about this process at `https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-convert-app-to-be-multi-tenant`.

In this section, we reconfigured the application to work as a multi-tenant app and we were victorious! Just note that you need to invite the Microsoft personal account as a guest user to work with your app. Next, we'll deploy another multi-tenant app with OpenID Connect to show you another implementation.

Deploying another multi-tenant app with OpenID Connect

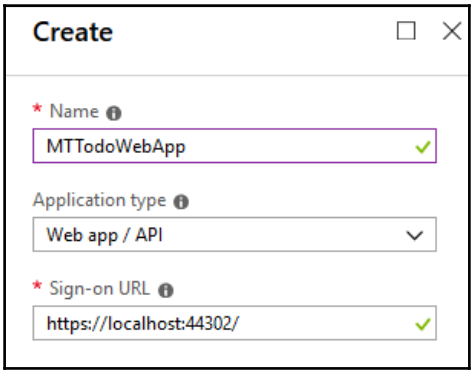
In this section, we'll install a multi-tenant app that works with OpenID Connect as an authentication protocol. Working through this sample will help you to deploy the correct app registration inside your Azure AD, and you'll learn what exactly needs to be configured in the application to use your Azure AD as an authentication provider:

1. Open the Azure portal: `https://portal.azure.com`.
2. Navigate to the Azure AD blade.
3. Click **App registrations**.
4. Click **+New application registration**:



Create a new app registration

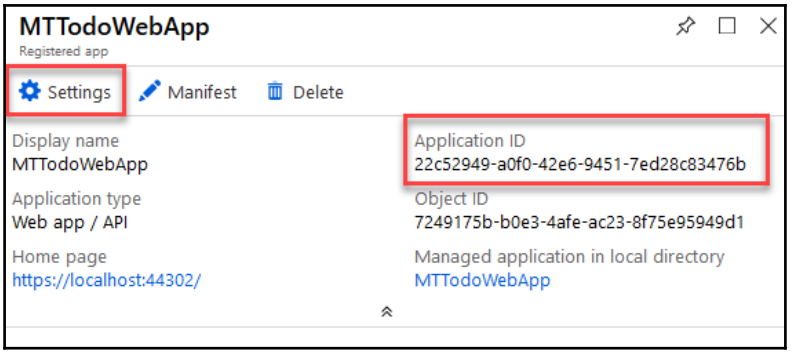
5. Provide an application name and the sign-on URL, `https://localhost:44302/`:



Provide the app properties

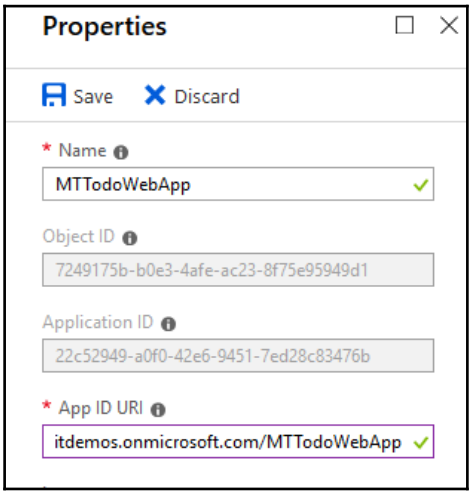
6. Copy the **Application ID** to Notepad.

7. Click **Settings**:



Get the app configuration data

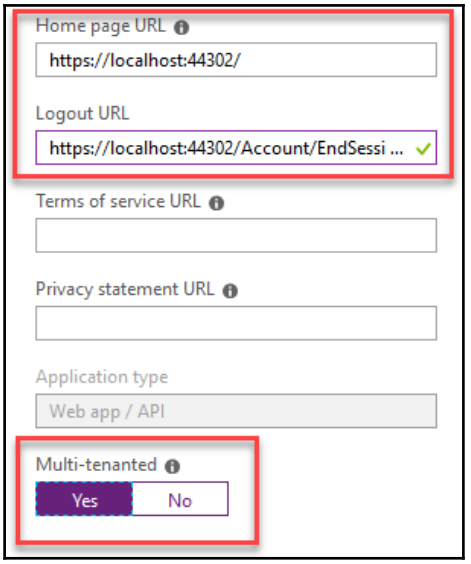
8. Provide the **App ID URI** in the format `https://181031inovitdemos.onmicrosoft.com/MTTodoWebApp`:



Adding the App ID URI

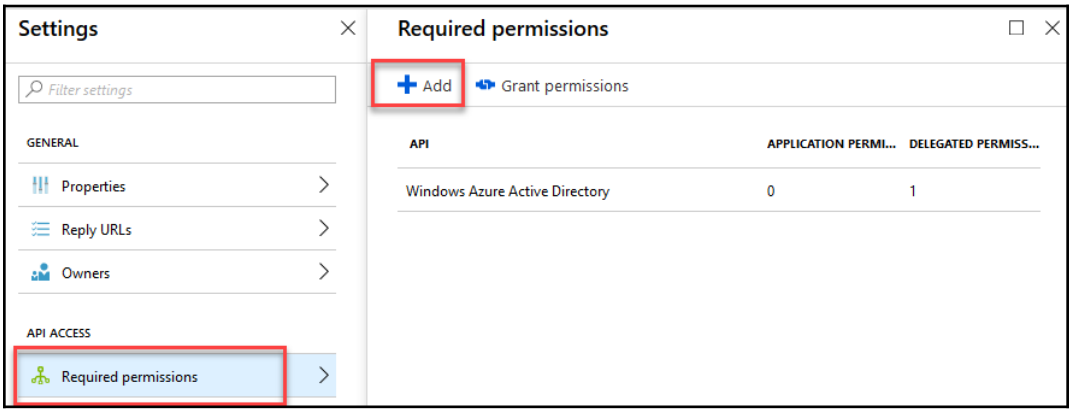
9. Provide the **Logout URL**. The **Home page URL** should already be filled in.
10. Provide the URL in the format `https://localhost:44302/Account/EndSession`.

11. Click **Multi-tenanted**: **Yes** to use the app as a multi-tenant application:



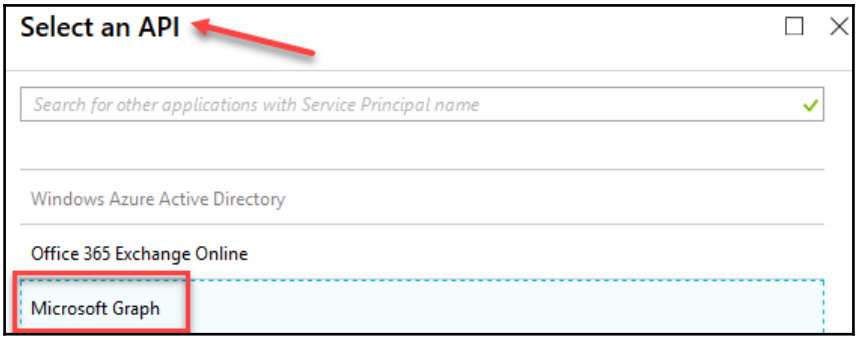
Using the multi-tenant option

12. Configure the correct permissions for the application. Click **Add** in the **Required permissions** section:



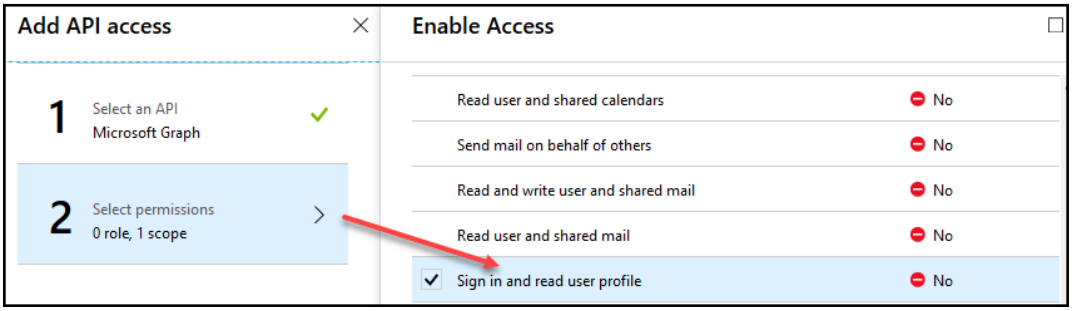
Grant the required permissions

13. Choose the **Microsoft Graph** API:



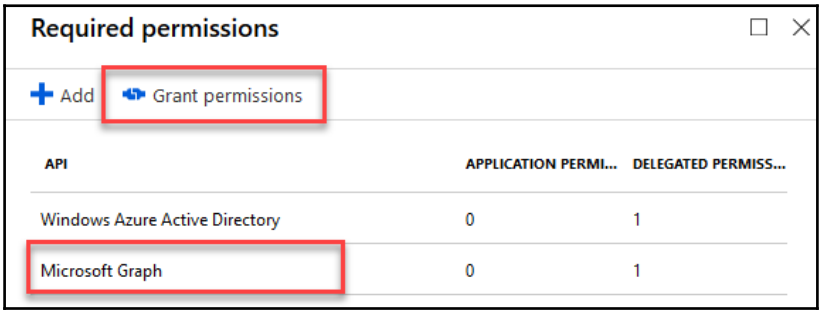
Use the Microsoft Graph API

14. Assign the **Sign in and read user profile** permission:



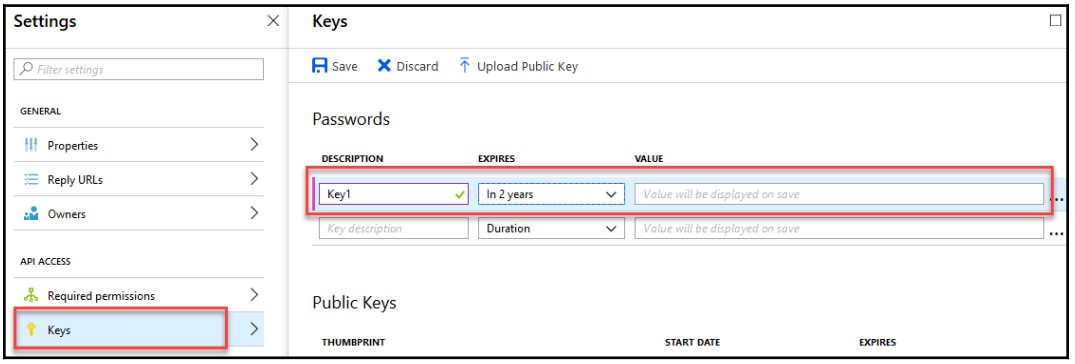
Scope assignment

15. Click **Grant permissions** that you grant the permissions:



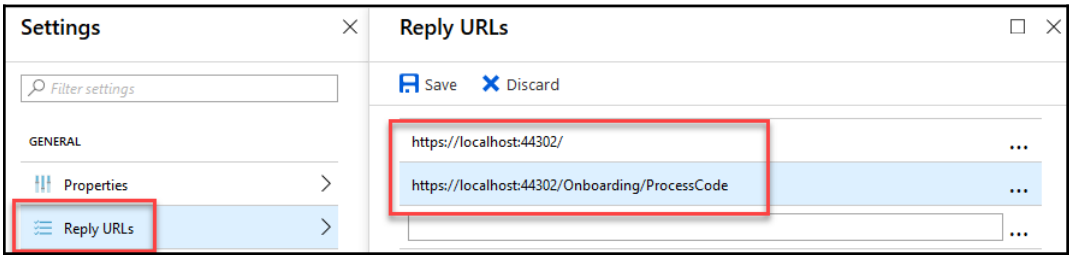
Grant permission dialog

16. Generate a key and call it `Key1`, with a duration of 2 years.
17. Click **Save** and copy the generated key value to Notepad:



Secret key generation

18. Provide the **Reply URLs**.
19. The following values need to be configured:
    - `https://localhost:44302/`
    - `https://localhost:44302/Onboarding/ProcessCode`
20. Here's the configuration:



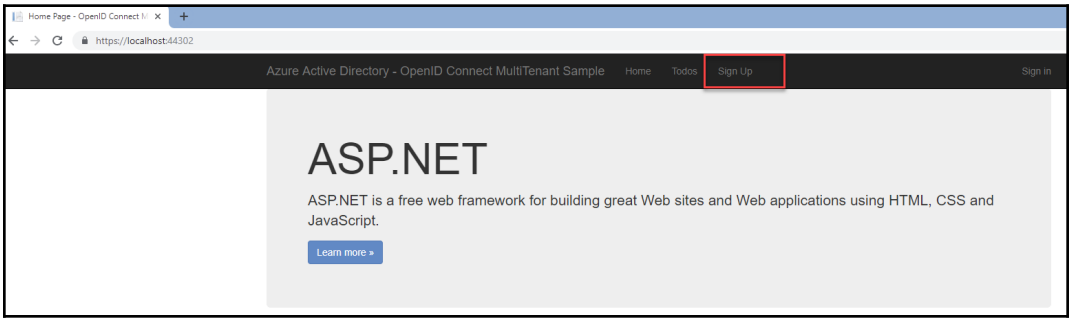
Reply URLs configuration

21. Open Visual Studio and the `WebApp-MultiTenant-OpenIDConnect-DotNet.sln` solution file.
22. Open the `web.config` file of the solution.
23. Copy the `Key1` value to the `ida:Password` section.
24. Copy the application ID to the `ida:ClientID` section.
25. The modified code should look like this:

```
<appSettings>
  <add key="webpages:Version" value="3.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  <add key="ida:ClientID" value="22c5294[redacted]476b"/>
  <add key="ida:Password" value="Jh8SAPS[redacted]yOhVyVHT8ZQ="  />
</appSettings>
```

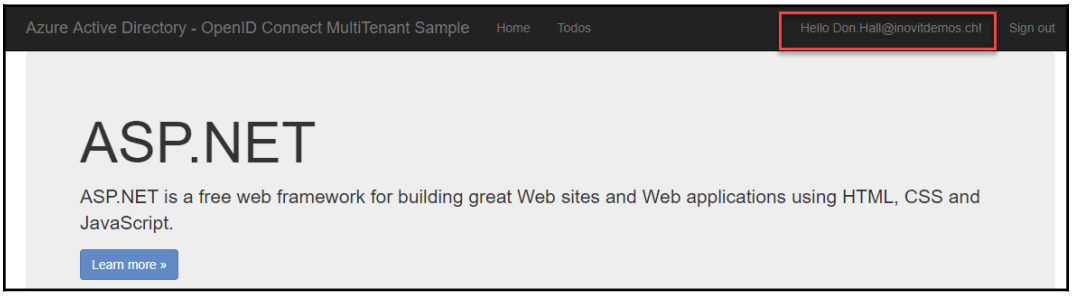
Code modification result

26. Test the multi-tenant functionality. Press *F5* in Visual Studio to build the solution.
27. The application should start.
28. Click **Sign Up**:



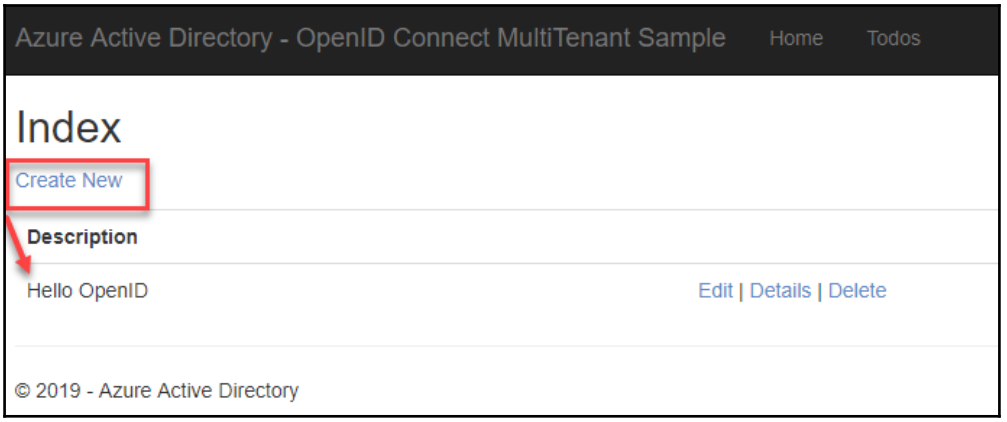
App Sign up process

29. **Sign Up** with the user Don Hall to see the expected behavior.
30. You are logged in, as you can see:



Successfully logging in to the application

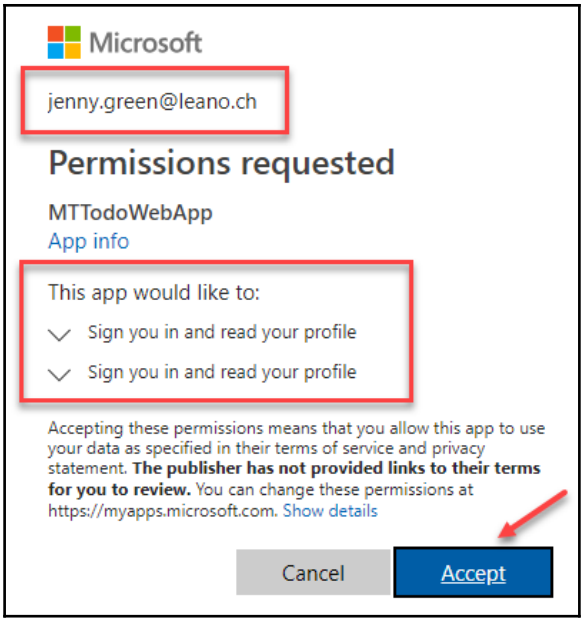
31. Create a test item to test the app's functionality. This isn't the critical part; we focus on the authentication:



Task creation dialog

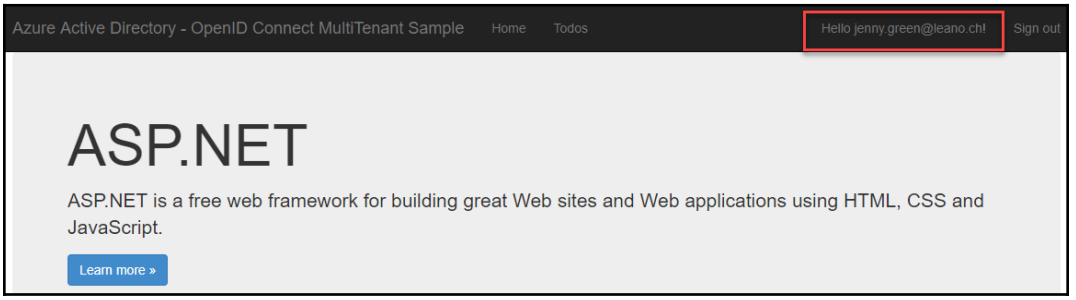
32. Try to sign up with your an Azure B2B guest user; you'll need to consent as a user to the app.

33. Use the admin consent in the sign-up process:



Checking and accepting the OAuth consent

34. **Accept** the consent, and you should be logged in:

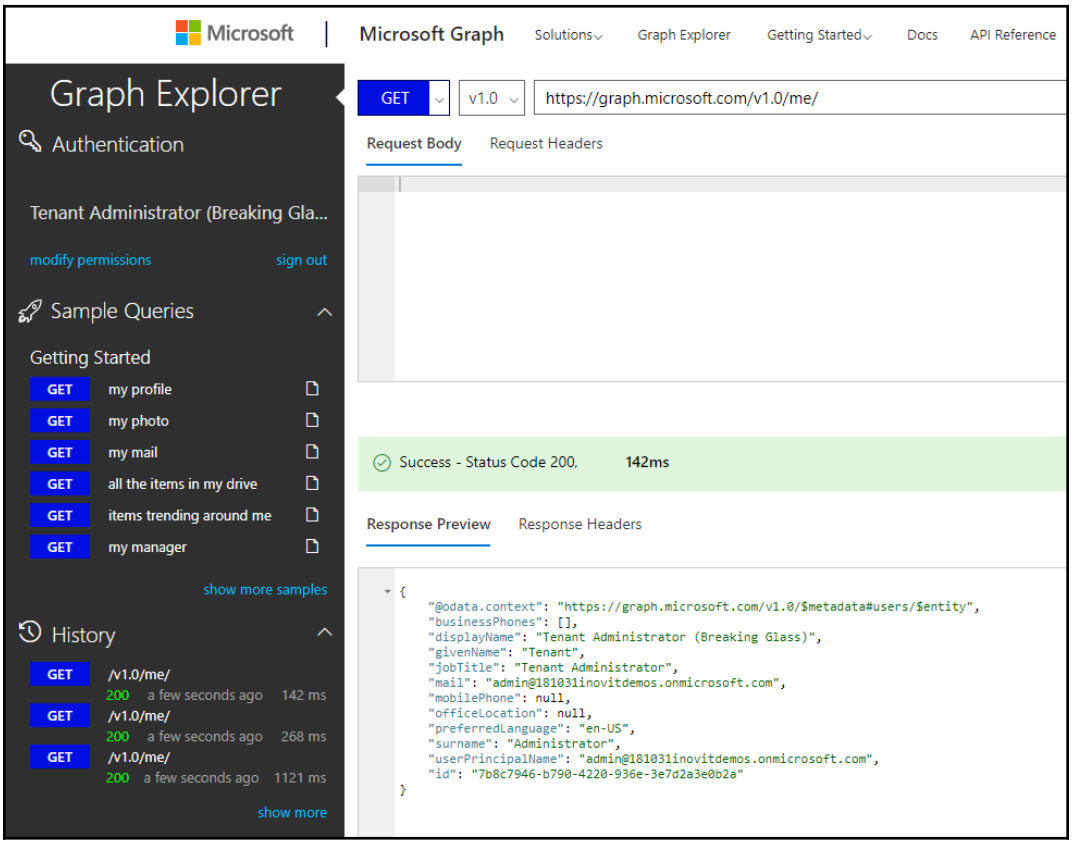


Successfully logging in to the application

35. Test the application with another user from any Azure AD tenant and you'll get the same experience.

Well done! We also see with this multi-tenant application and OpenID Connect implementation.

A final tip: configure the **Graph Explorer** for your tenant to work with the Microsoft Graph API, and gather more information about this topic. Visit `https://developer.microsoft.com/en-us/graph/graph-explorer` and log in as a global admin to your Azure AD:



Microsoft Graph Explorer application

With all this experience, you can now start analyzing and extending the solutions.

Summary

In this chapter, you learned about single- and multi-tenant applications and the differences between them. You handled both application types in your environment. Furthermore, you discovered roles and claims in both application types, so now you can provide a **role-based access control** (**RBAC**) scenario. With all the other applications you've deployed, you now understand the differences between the different protocols, such as SAML2.0, OAuth2 with the different flow types, and OpenID Connect. You also learned how the **System for Cross-domain Identity Management** (**SCIM**) helps you to provision users (synchronizing) from Azure AD to the application.

In the next chapter, you'll learn how to provide a cloud-based identity management life cycle. We'll focus on how to provide secure and usable authentication and identity management to your users.

10
Exploring Azure AD Identity Services

In this chapter, we'll explore the different Azure AD identity services and AD FS as an on-premise identity service. We'll look at the Azure AD B2B and B2C functionality and explain the main concepts behind these technologies. Furthermore, we'll look at and extend the Azure AD Domain Services that we configured in `Chapter 1`, *Building and Managing Azure Active Directory*. To get the whole picture, we'll also view the different capabilities of the Active Directory Federation Services, and how they support different authentication scenarios. You will learn how to use the Azure AD B2B and B2C services for your projects, in order to provide suitable access to your applications for customers, partners, and internal employees. Particularly, you can use Azure AD B2C as a complete identity platform for your developed applications.

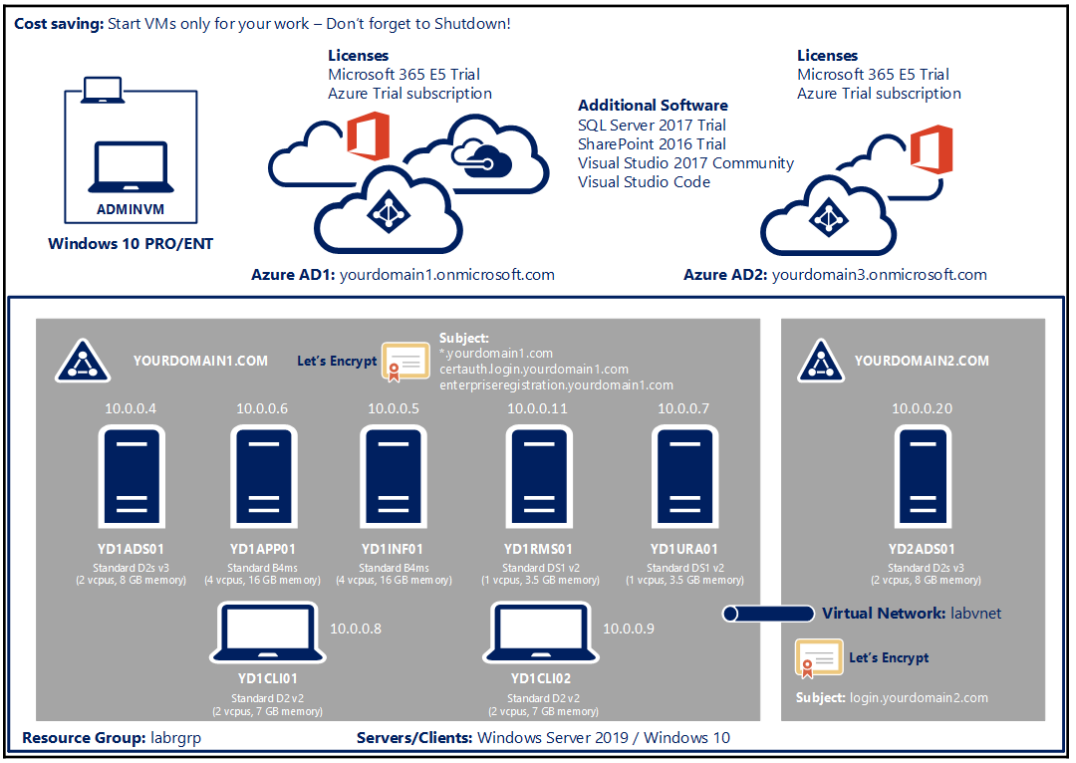
The chapter will be divided into the following sections:

- Preparing your lab environment
- Understanding Azure AD **Business to Business** (**B2B**)
- Exploring Azure AD **Business to Customer** (**B2C**)
- Extending Active Directory solutions with Azure AD Domain Services
- AD FS as an on-premise identity service for the cloud

In the first section, we start preparing our lab environment.

Preparing your lab environment

In this chapter, we will need to have an additional Azure AD, configured with Office 365, to test the different features. You already know how to create this configuration from `Chapter 1`, *Building and Managing Azure Active Directory*. Build this additional tenant with a minimum set of configurations. Basically, you'll need to have a custom domain verified and registered, and nothing else. Furthermore, you will need to install Visual Studio 2017 Community with the ASP.NET and web development workload on your administrative workstation:



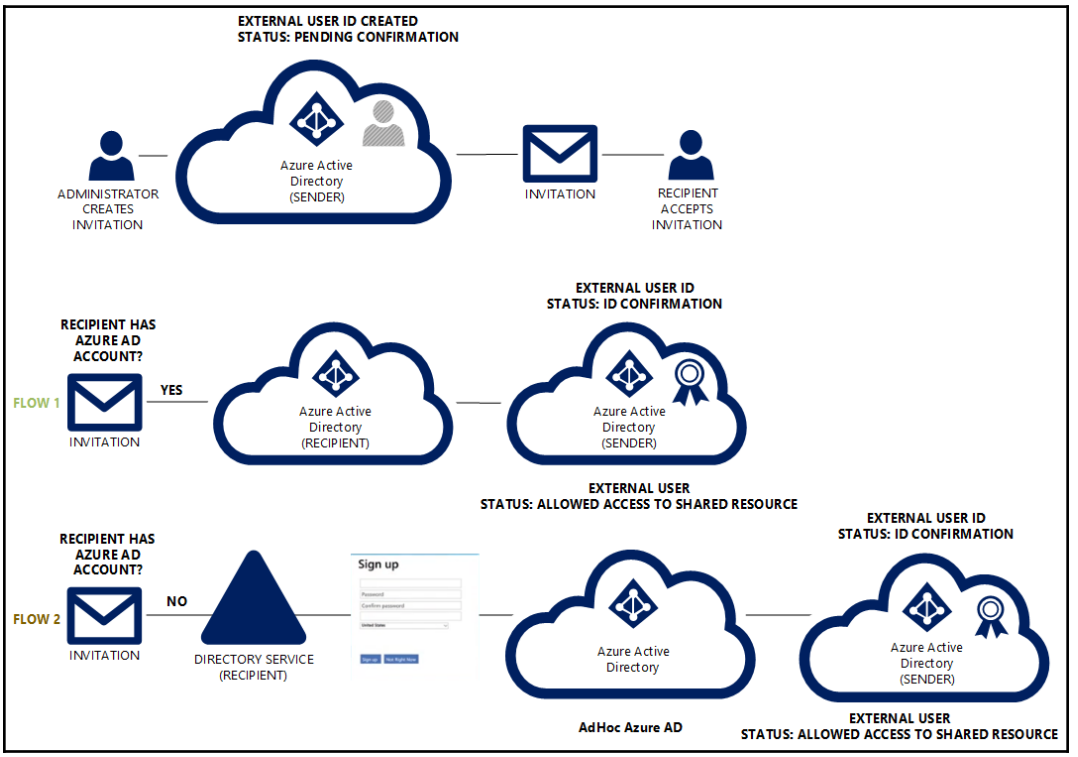
Lab environment overview

To configure the Azure AD Domain Services LDAPS use case, you need to provide a public SSL certificate. You can use the procedure from `Chapter 7`, *Deploying Solutions on Azure AD and AD FS*, to generate certificates for your needs with Let's Encrypt solution. We'll start with the Azure AD B2B functionality.

Understanding Azure AD B2B

Azure AD B2B solves the problem of collaboration between business partners. It allows users to share business applications between partners, without going through inter-company federation relationships and internally-managed partner identities. With Azure AD B2B, you can create cross-company relationships by inviting and authorizing users from partner companies to access your resources. With this process, each company federates once, with Azure AD, and each user is then represented by a single Azure AD account. This option also provides a higher security level, because if a user leaves the partner organization, access is automatically disallowed. Inside of Azure AD, the user will be handled as a guest, and they won't be able to traverse other users in the directory. Permissions of the invited user will be provided over the correct associated group membership.

The following figure shows the process of enabling business partners to access your applications:



Azure AD B2B invitation flows

In the case of **FLOW 1**, the user will be able to sign in to the partner organization after they accept, and consent to, the invitation.

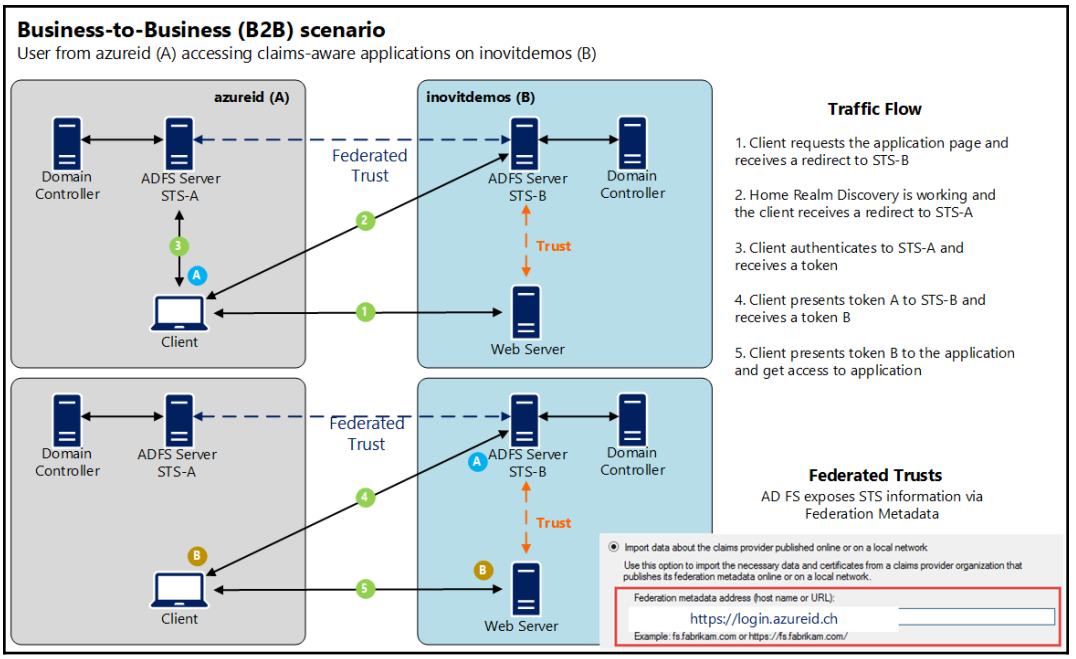
In the case of **FLOW 2**, the user signs up for their own Azure Active Directory and will be added to the Azure AD from which the invitation process was started.

You can find more information about the service at `https://docs.microsoft.com/en-us/azure/active-directory/b2b/what-is-b2b`. In `Chapter 11`, *Creating Identity Life Cycle Management on Azure*, we'll provide the configuration tasks for the complete guest-management life cycle, which includes the Azure AD B2B portal, using Azure MFA, conditional access, and access reviews.

Providing resource access to external partners (on-premise)

With the following on-premise configuration, we can provide resource access to external partners. You can use AD FS for this task, to allow employees and customers or partners to access your claims-aware applications. We can achieve the goal of providing federated B2B access by configuring federation partners.

The design and the traffic flow are shown in the following diagram:



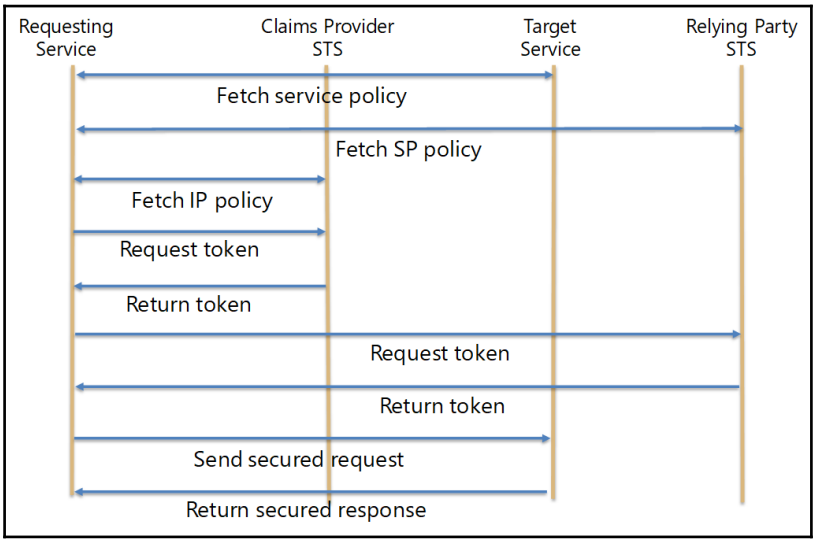
ADFS B2B configuration

Another capability of Active Directory Federation Service is the ability to support active clients using the **WS-Trust** specifications. This allows your client software to interact with the different players in such a scenario, without relying on browser redirects to locate the claims provider, resource provider, or other relevant components.

The flow runs in the following steps:

1. The **Requesting Service** (software on client) queries the **Target Service** and requests a list of policy requirements. This includes a list of required claims and the STS.
2. The Requesting Service queries the **Relying Party STS** for policies. This includes the list of claims providers that the STS trusts.

3. The Requesting Service queries the **Claims Provider STS** for the list of policies. This includes the required authentication method and other information.
4. The Requesting Service receives all of the policy information; the client will request a token from the **Claims Provider** (**CP**). This is a direct connection to the CP using **SOAP over HTTPS**.
5. The **CP** authenticates the user and returns a token.
6. The Requesting Service receives the token; a token will be directly requested from the **Relying Party** (**RP**) using SOAP over HTTPS.
7. The **Relying Party Federation** Server signs the token and sends it back to the user.
8. The **Requesting Service receives** the token that has been issued and signed by the **Relying Party STS**. It submits the token to the target services and receives a response:



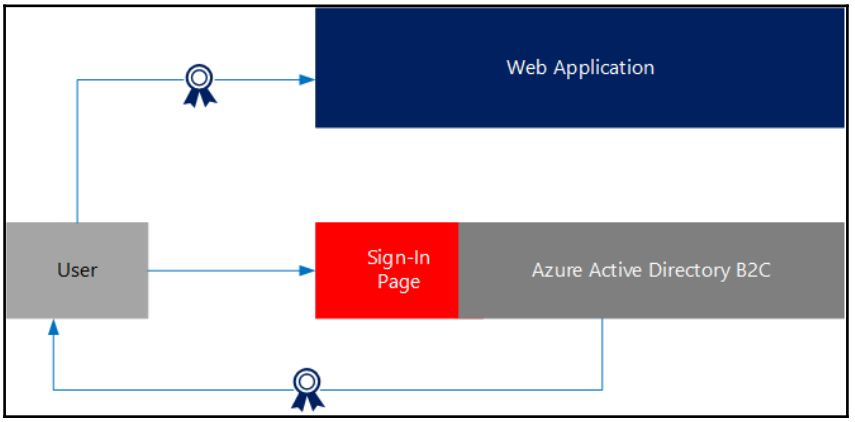
WS-trust flow

Remember that this on-premise extension can always be supported and used with Azure Identity and Access Management services.

Exploring Azure AD B2C

Azure AD B2C builds a complete identity-management framework for developers and supports signing in to your application using social networks, such as Facebook, Google, or LinkedIn, and creating developed accounts with usernames and passwords specifically for your company-owned application. Self-service password management and profile management are also provided. Additionally, Azure MFA introduces a higher grade of security to the solution. Principally, this feature allows for small, medium, and large companies to hold their customers in a separate Azure Active Directory, with all the capabilities, and more, in a similar way to the corporate-managed Azure Active Directory. With different verification options, you are also able to provide the necessary identity assurance required for more sensible transactions. Azure AD B2C takes care of all of the IAM tasks for your own development activities.

Basically, the minimum architecture with the usage of Azure AD B2C looks like the following example. As we already mentioned, Azure AD B2C provides the identity-management framework for your application:



Azure AD B2C basic concepts

To get a better understanding of Azure AD B2C, we'll build an example application from Microsoft that provides a small web application, including a web API. This application is a good starting point to get deeper into Azure AD B2C. We highly recommend building it in your lab environment. The application can also run against a predefined demo environment. You will find the source at `https://github.com/Azure-Samples/active-directory-b2c-dotnet-webapp-and-webapi#Using-the-demo-environment`.

We recommend working through the following Azure AD B2C introduction before you start the lab activities: `https://docs.microsoft.com/en-us/azure/active-directory-b2c/active-directory-b2c-overview`

Let's start our journey and log in to our administrative workstation.

We'll do all of the Azure AD B2C tasks on the first full-blown Azure AD that we created.

In the next section, we will create the Azure AD B2C tenant.

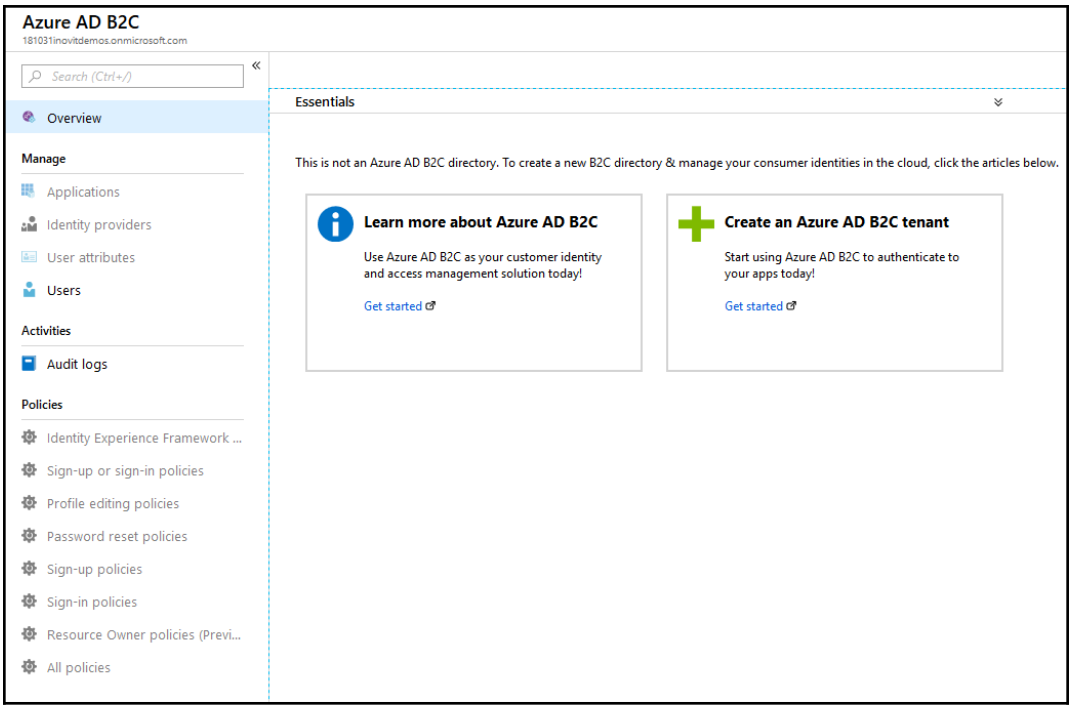
Azure AD B2C tenant creation

Now that we know the basic concept of Azure AD B2C, we will start our configuration for the test application that uses Azure AD B2C to provide the identity platform:

1. Open a shell or command line in your Visual Studio project folder, in my case `C:\Users\jochen.nickel\Documents\Visual Studio 2017\Projects`, and execute the following command:

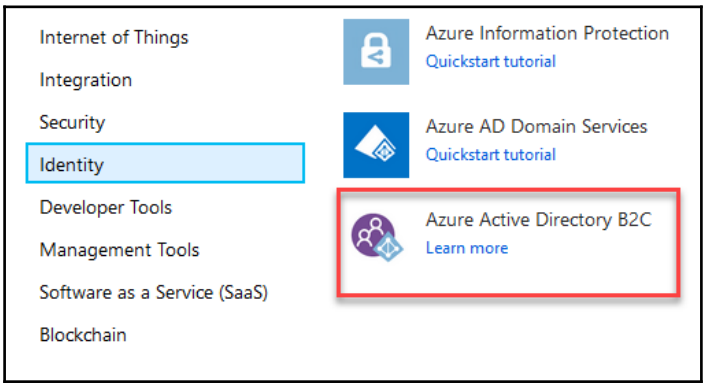
```
git clone
https://github.com/Azure-Samples/active-directory-b2c-dotnet-webapp-and-webapi.git
```

2. Create the **Azure AD B2C directory**.
3. Open the Azure Portal (`https://portal.azure.com`) with global administrator credentials.
4. Use the search option to find the **Azure AD B2C** blade to view the service that we'll create:



Azure AD B2C creation procedure

5. Click on **Create a resource** in the top-left corner of the Azure portal.
6. Click on **Identity** and choose **Azure Active Directory B2C**:

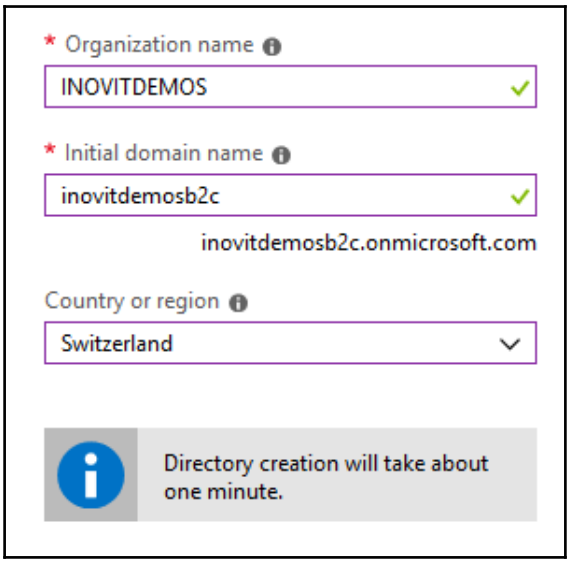


Choose Azure AD B2C

7. Click on **Create a new Azure AD B2C Tenant**:

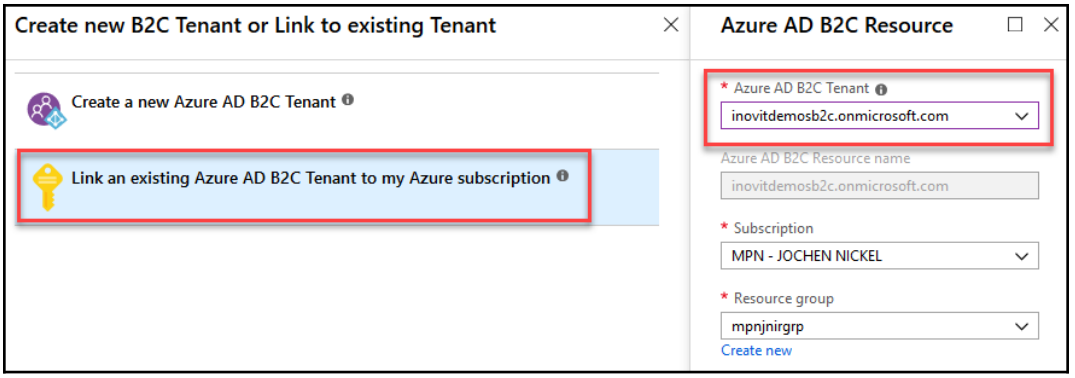
Create the new Azure AD B2C tenant

8. Use the `YOURDOMAIN1` and `YOURDOMAINB2C` values for your demo environment:



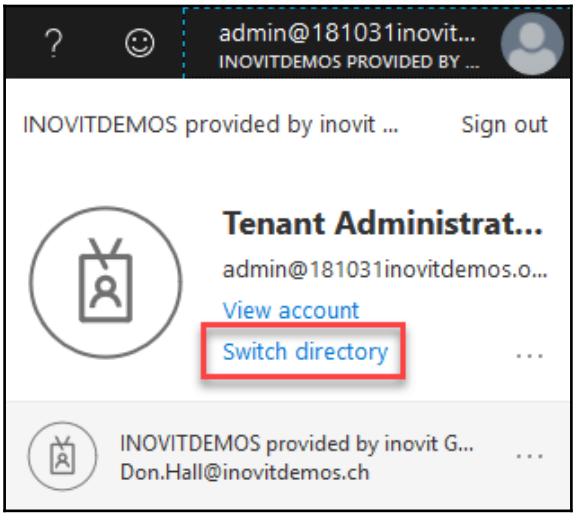
Azure AD B2C properties

9. Link the new Azure AD B2C to our Azure subscription:



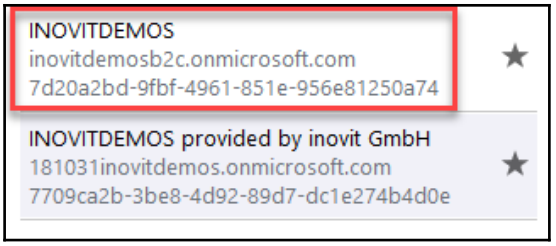
Linking the Azure AD B2C tenant

10. Switch the directory to use the new Azure AD B2C tenant:



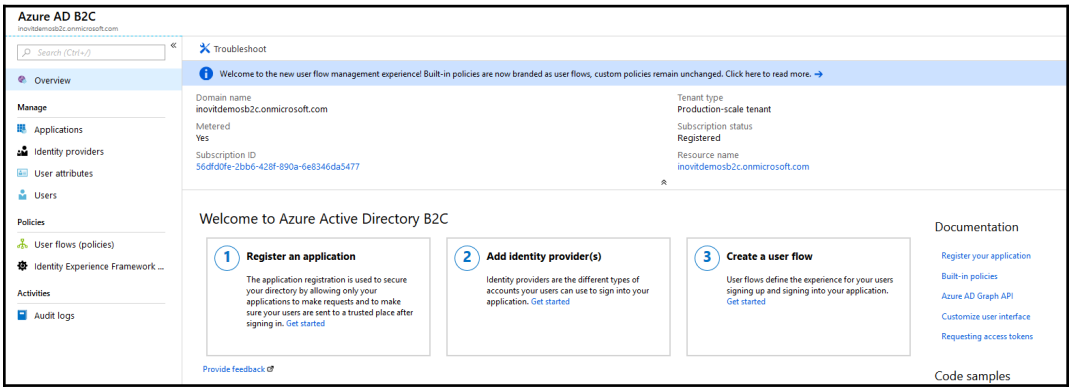
Azure AD directory switching

11. Use the Azure AD B2C directory:



Choose the newly created Azure AD B2C tenant

12. Navigate to the **Azure AD B2C** blade:



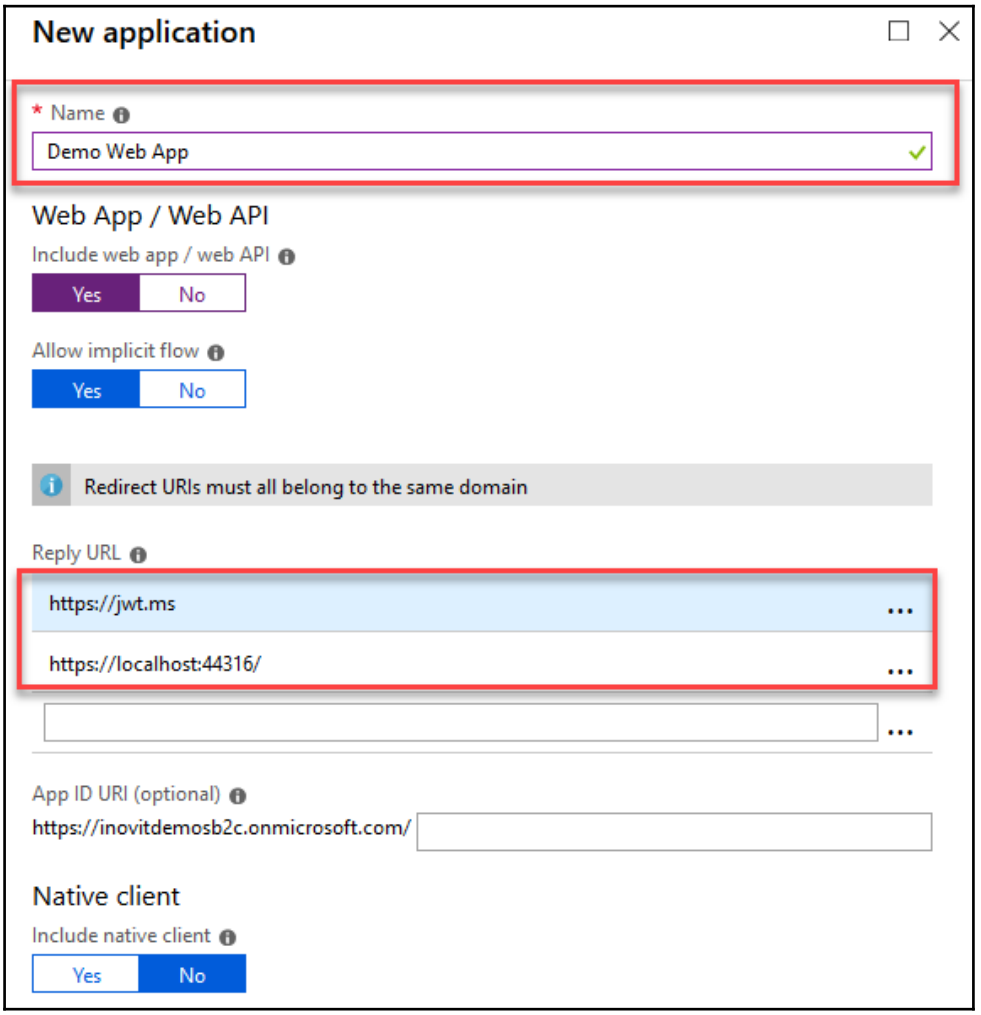
Azure AD B2C overview page

In the next section, we will configure the demo application in Azure AD.

Demo app registration

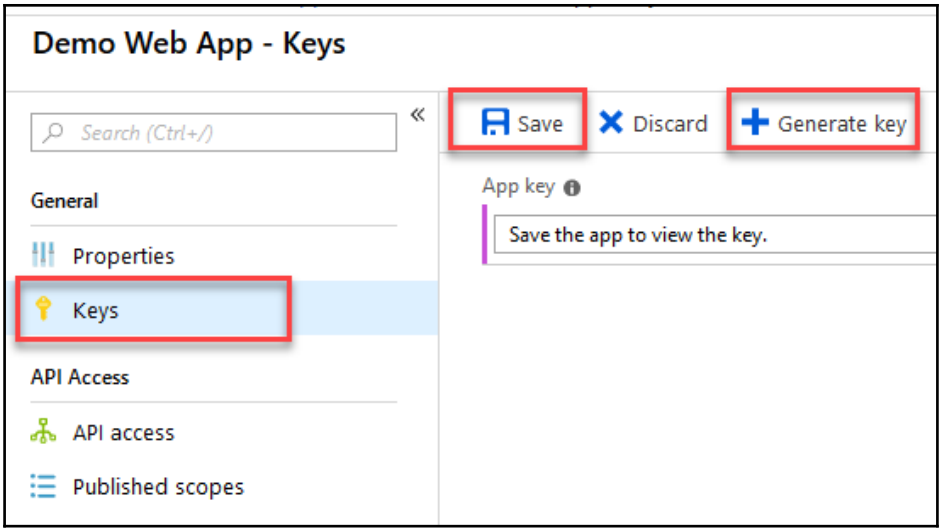
Next, we'll register the new `Demo Web App` and the **Web App** API under **application**:

1. Add the **web app**, as shown in the following screenshot:



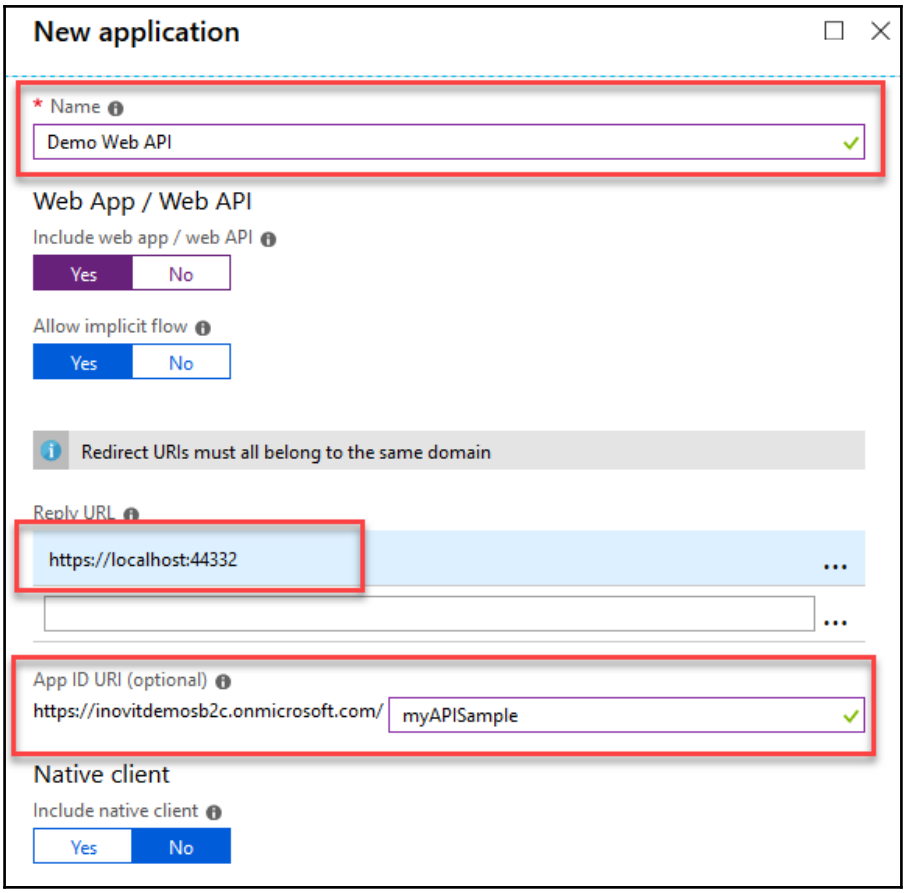
Provide the app properties

2. Create an app key:



App key generation process

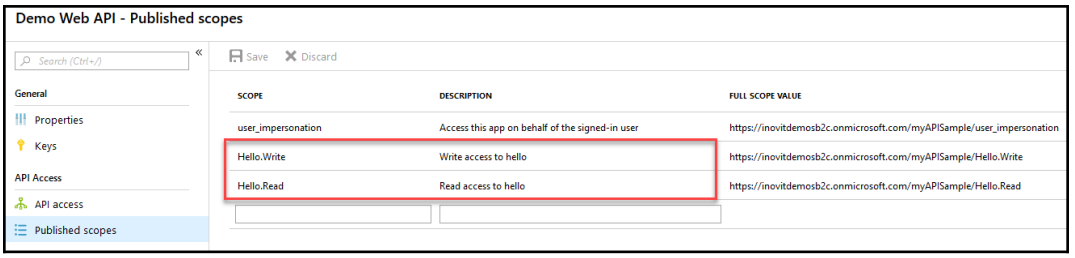
3. Copy the key value to a notepad.
4. Copy the app ID under the properties section to the notepad.
5. Register the second app for the **Web App** API.
6. Fill in the values, as shown in the following screenshot:



Register the second application with properties

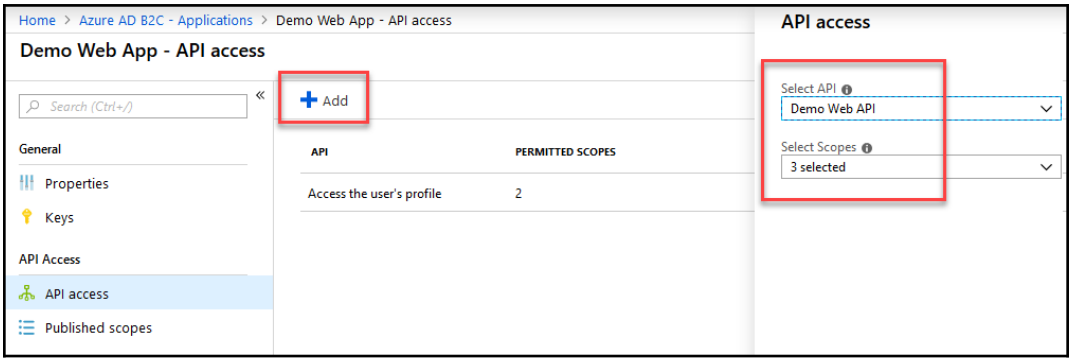
7. Copy the **App ID URI (optional)** to the notepad.

8. Configure the **Published scopes** of the web API:
    - **Hello.Write | Write access to hello**
    - **Hello.Read | Read access to hello**:



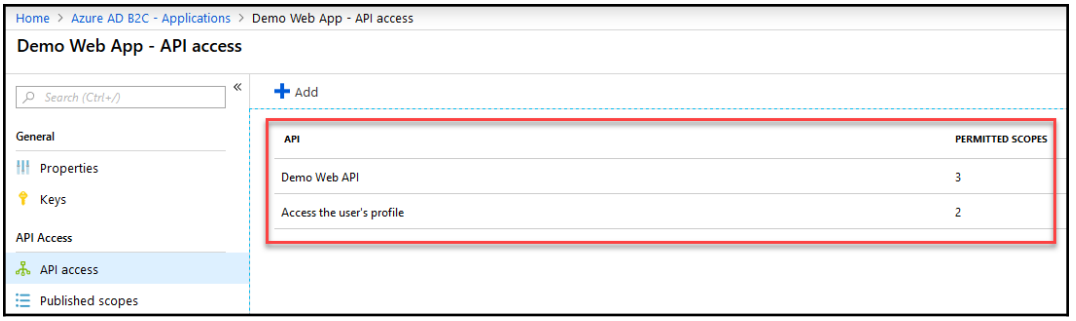
Defining scopes

9. Change to the **Web App** and **API access** section.
10. Click on **Add** and select the **Demo Web API**:



Define the app access

11. You should be presented with the following:



API access result

12. You should have a notepad with the following values:

```
Web App
ID d92c671f-c576-45b1-9d53-0510f306bccc
Key WdLrw#+DZ6L#vW7,3(ryS.uP

Web App API
ID 2f73f192-e9c5-4fa8-8429-9aa3fd13293e
```

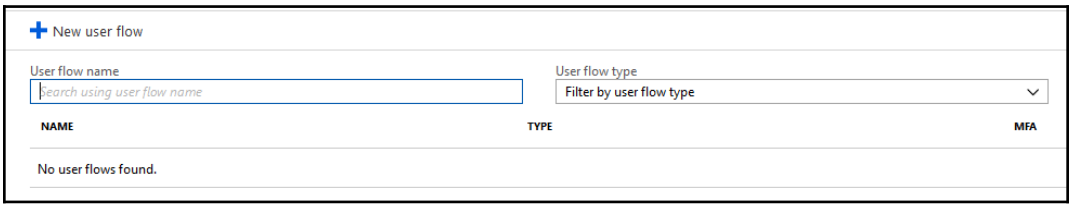
Now, we can start the user flow creation in Azure AD B2C.

User flow creation

In the following section, we will configure the first user flow in Azure AD B2C to get a user sign-up process to the demo application.

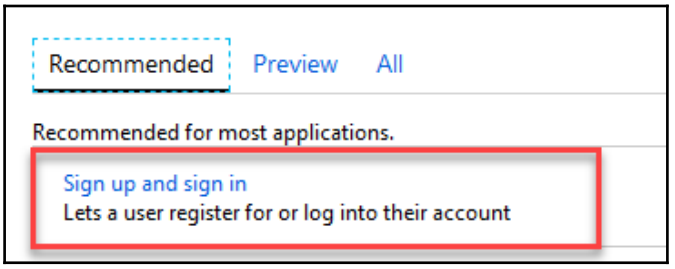
Next, we will create our first user flow:

1. Navigate to **user flows** (policies) and create your first flow:



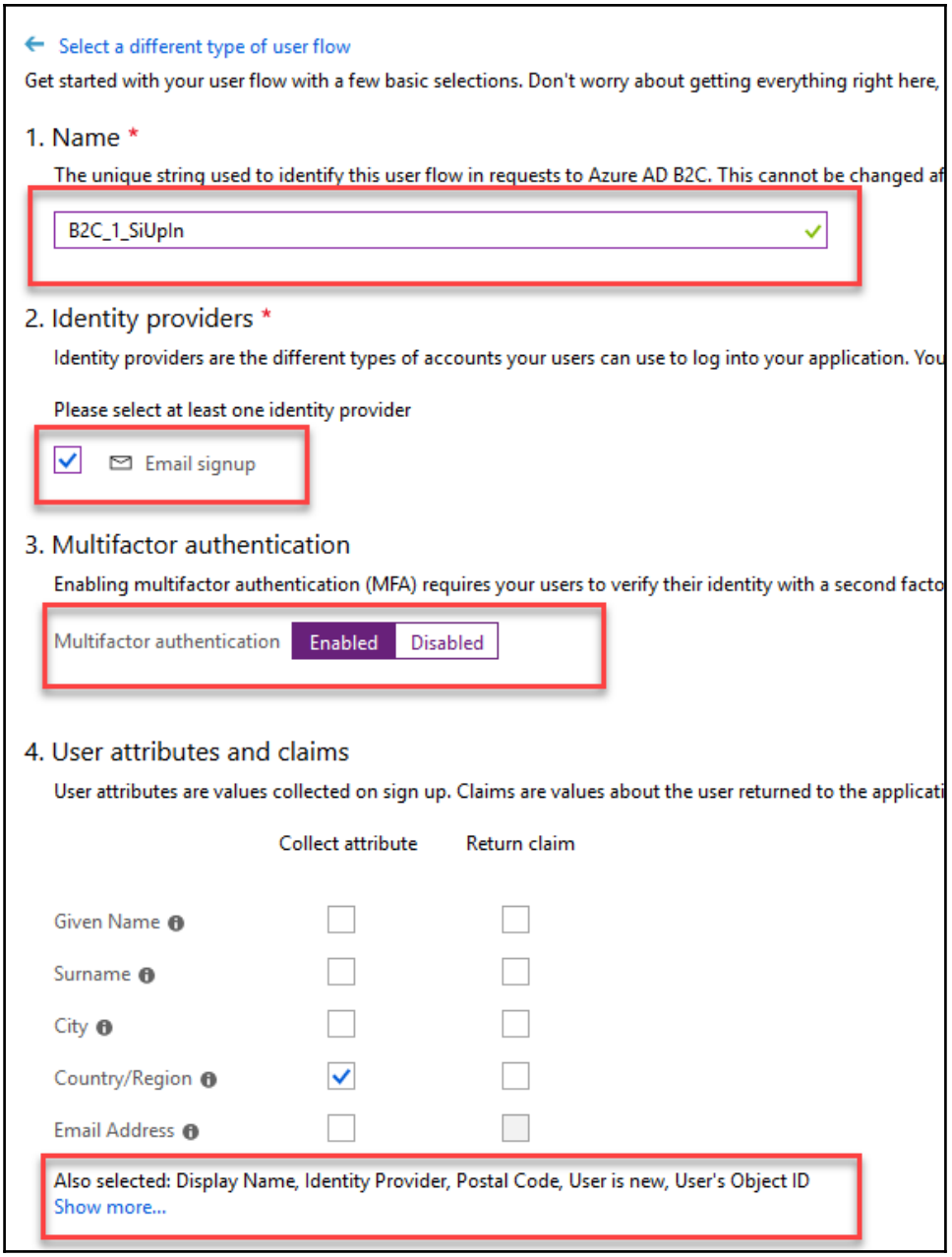
New user flow creation

2. Use the **Sign up and sign in** option:



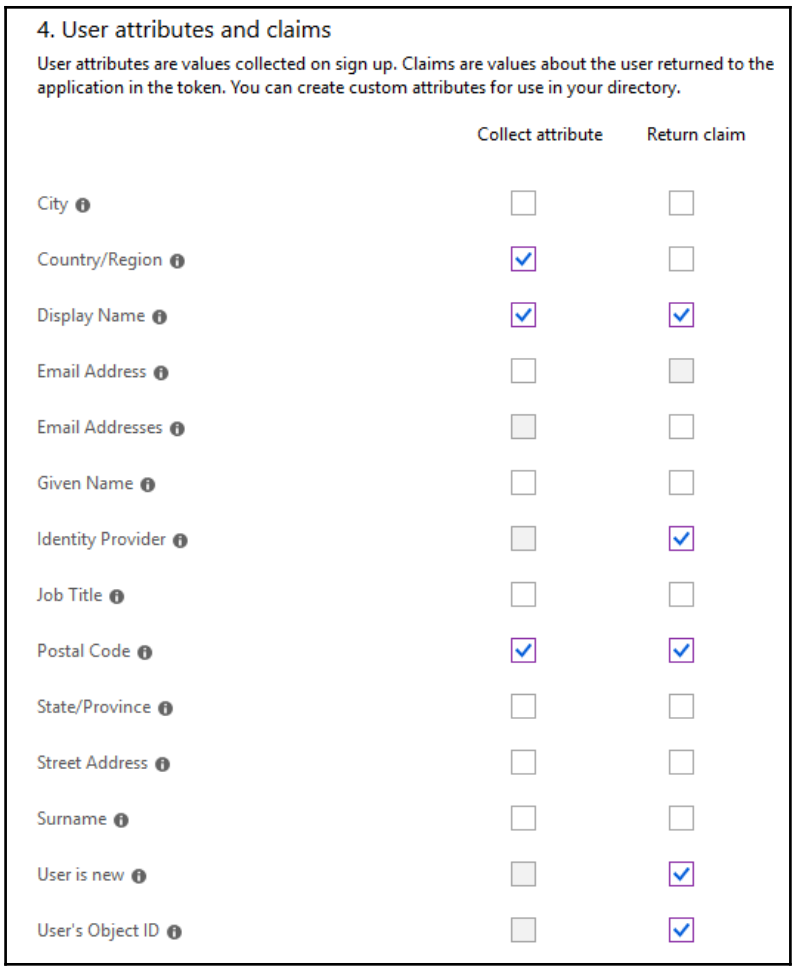
Choosing the Sign up and sign in option

3. Use the following values for the flow:



Defining the flow properties

4. And the following attributes:



Choosing the attributes that will be collected and will also return a claim

5. Click on **Create.**

Now that we have prepared the **Sign up and sign in** options, we can start to modify the demo application code.

Visual Studio code modification

To use the newly created user flow, we need to modify the demo application code.

Modify the `TaskWebApp` project, as follows:

1. Open the `web.config` file from `TaskWebApp`.
2. Find the following keys and replace them:
    - `ida:Tenant`, with your tenant name, `yourdomainb2c.onmicrosoft.com`
    - `ida:ClientId`, with the app ID from the notepad
    - `ida:ClientSecret`, with your key from the notepad
    - `ida:SignUpSignInPolicyId`, with `b2c_1_SiUpIn`
3. Comment out the following entries:

```
<!--<add key="api:TaskServiceUrl"
value="https://aadb2cplayground.azurewebsites.net/" />-->
```

4. Uncomment the following entries:

```
<add key="api:TaskServiceUrl"
value="https://localhost:44332/"/>
```

5. Change the `api:ApiIdentifier` key value to the app ID URI of the API:

```
<!--<add key="api:ApiIdentifier"
value="https://fabrikamb2c.onmicrosoft.com/api/" />—->
<add key="api:ApiIdentifier"
value="https://yourdomainb2c.onmicrosoft.com/myAPISample/"
/>
```

6. Your code should look as follows:

```
<configuration>
  <appSettings>
    <add key="webpages:Version" value="3.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
    <add key="ida:Tenant" value="inovitdemosb2c.onmicrosoft.com" />
    <add key="ida:ClientId" value="d92c671f-c576-45b1-9d53-0510f306bccc" />
    <add key="ida:ClientSecret" value="WdLrw#+DZ6L#vW7,3(ryS.uP" />
    <add key="ida:AadInstance" value="https://login.microsoftonline.com/tfp/{0}/{1}/v2.0/.well-known/openid-configuration" />
    <add key="ida:RedirectUri" value="https://localhost:44316/" />
    <add key="ida:SignUpSignInPolicyId" value="b2c_1_SiUpIn" />
    <add key="ida:EditProfilePolicyId" value="b2c_1_SiPe" />
    <add key="ida:ResetPasswordPolicyId" value="b2c_1_SSPR" />
    <!-- Uncomment the localhost url if you want to run the API locally -->
    <add key="api:TaskServiceUrl" value="https://localhost:44332/" />
    <!--add key="api:TaskServiceUrl" value="https://localhost:44332/" /-->
    <!-- The following settings is used for requesting access tokens -->
    <add key="api:ApiIdentifier" value="https://inovitdemosb2c.onmicrosoft.com/myAPISample/" />
    <add key="api:ReadScope" value="Hello.Read" />
    <add key="api:WriteScope" value="Hello.Write" />
  </appSettings>
```

Modified code reference

Modify the `TaskService` project, as follows:

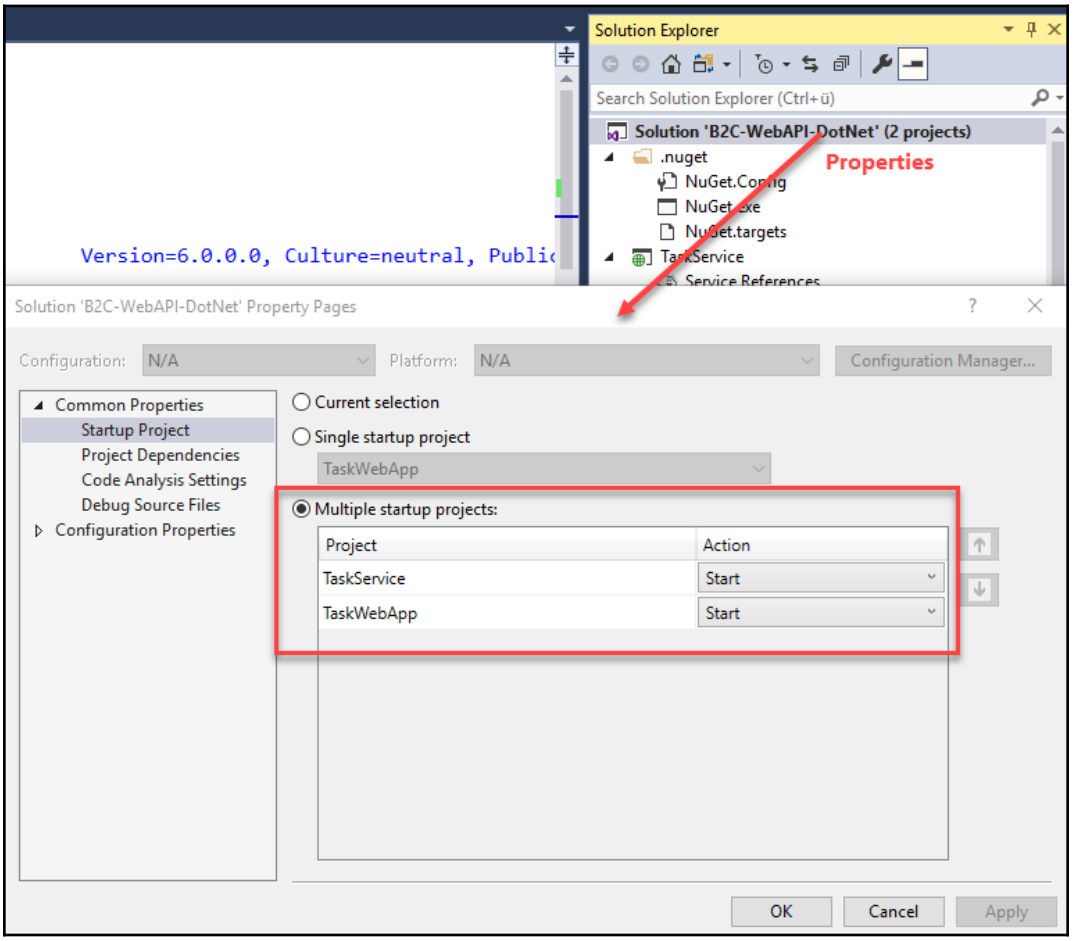
1. Open the `Web.config` file from the `TaskService` project.
2. Find the following keys and replace them:
    - `ida:Tenant`, with your `yourdomainb2c.onmicrosoft.com`
    - `ida:ClientId`, with the application ID from your web API
    - `ida:SignUpSignInPolicyId`, with `b2c_1_SiUpIn`
    - `api:ReadScope`, with `Hello.Read`
    - `api:WriteScope`, with `Hello.Write`
3. Your code should look like this:

```
<appSettings>
  <add key="webpages:Version" value="3.0.0.0" />
  <add key="webpages:Enabled" value="false" />
  <add key="ClientValidationEnabled" value="true" />
  <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  <add key="ida:AadInstance" value="https://login.microsoftonline.com/{0}/v2.0/.well-known/openid-configuration?p={1}" />
  <add key="ida:Tenant" value="inovitdemosb2c.onmicrosoft.com" />
  <add key="ida:ClientId" value="2f73f192-e9c5-4fa8-8429-9aa3fd13293e" />
  <add key="ida:SignUpSignInPolicyId" value="B2C_1_SiUpIn" />
  <!-- The following settings is used for requesting access tokens -->
  <add key="api:ReadScope" value="Hello.Read" />
  <add key="api:WriteScope" value="Hello.Write" />
</appSettings>
<!--
```

Modified code reference

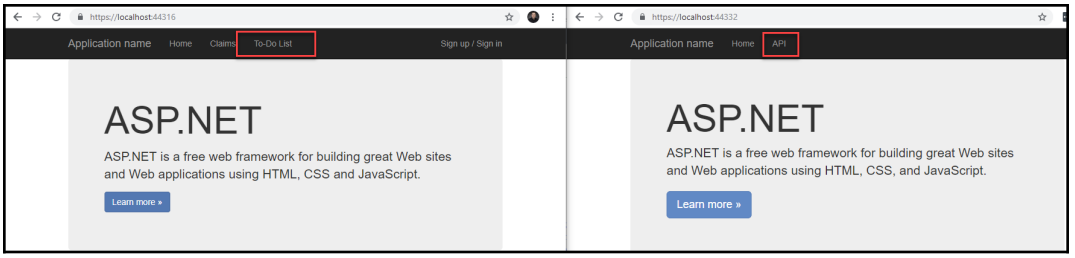
The Visual Studio **Startup Project** modification is as follows:

1. Go to the **Solution Explorer** and right-click on **Properties**.
2. Under **Common Properties**, go to **Startup Project**.
3. Use the following configuration:



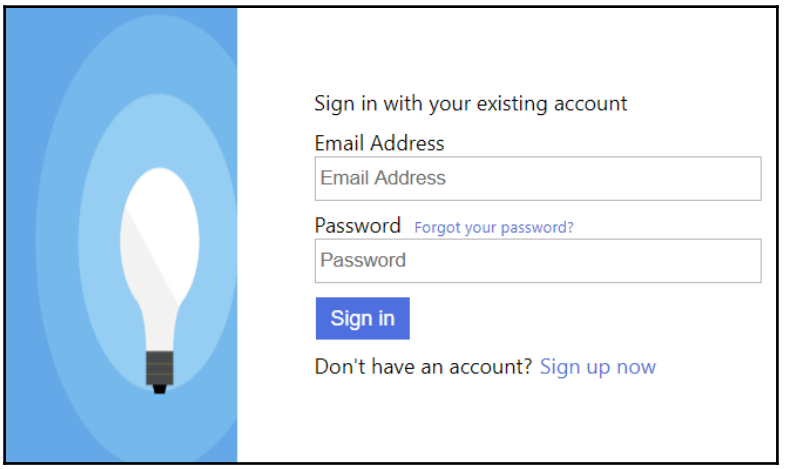
Configuring the project start up options

4. Press *F5* in Visual Studio to start both the Web App and the Web App API:



Started applications view

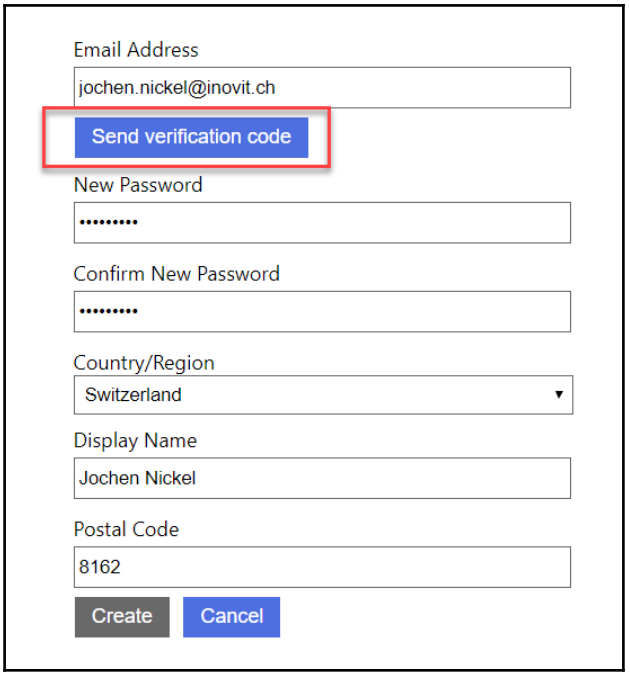
5. Click on **Sign up/Sign In** on the web app, and Azure AD B2C will come into the game:



Testing the user flow

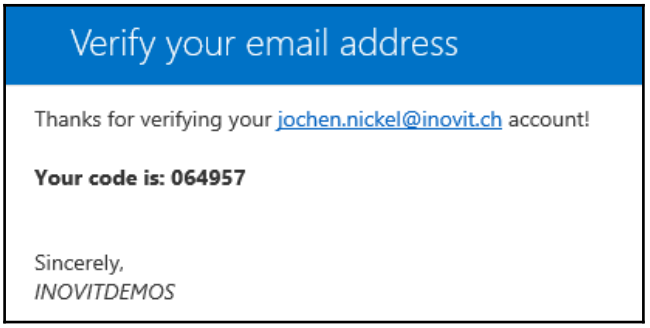
6. Click on **Sign up now**.

7. Enter your details and click on **Send verification code**:



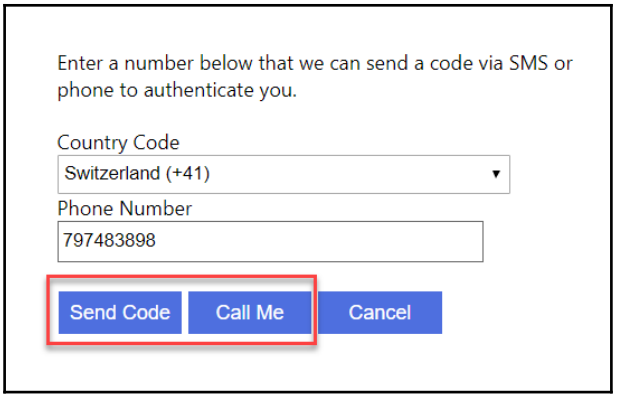
Providing the required data

8. You'll receive an email with the verification code, as follows:



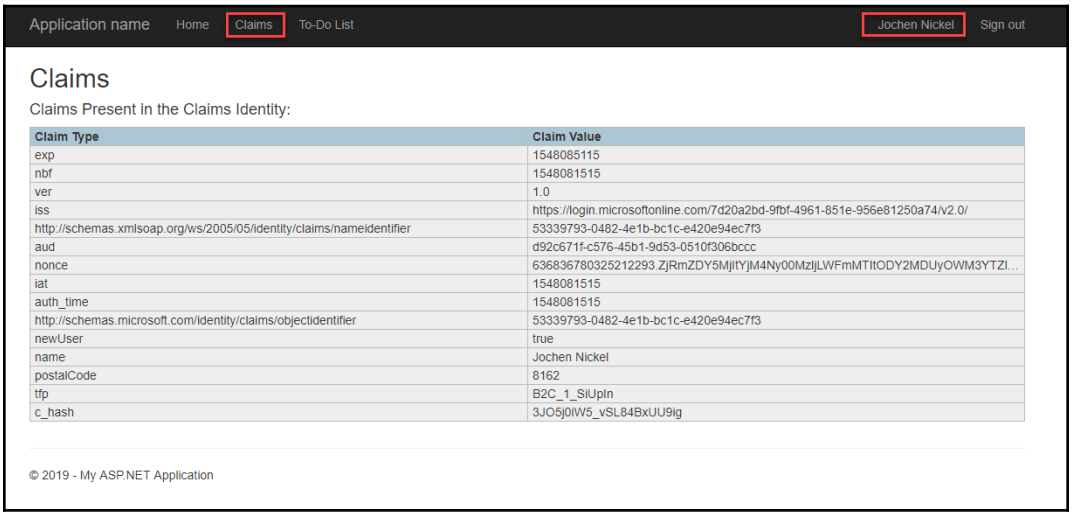
Get your verification code

9. Verify and create the account.
10. Azure MFA will now come into the game; provide your preferred verification option:



Providing your verification

11. You've successfully signed in, and you can view your claims:



Successful login on the demo app, including the provided claims

Yeah! Well done. You can dive deeper into Azure AD B2C at `https://docs.microsoft.com/en-us/azure/active-directory-b2c/`.

Comparing Azure AD B2B and B2C

Basically, both of the Azure AD services allow you to work with external users. Azure AD B2B focuses on the business to simplify the collaboration process with the secure sharing of information and resources. It takes care of the federation between the organizations and allows for a simple invitation-and-redemption process with different account types, such as school and work accounts, or simply an email account.

Azure AD B2C focuses on developers that create customer-facing apps. Developers get a full-featured identity system for their applications. Azure AD B2C provides a local repository and the sign-in experience with many other identity providers:

Azure AD B2B	**Azure AD B2C**
Authentication of users from partner organizations	Customer access with mobile and web apps
Partner life cycle = host and inviting organizations, including access reviews	Customer life cycle = self-service or application-managed
Supports work or school accounts, or any email address	Supports local user application accounts or supported identity providers
Supports SSO to all Azure AD connected applications	Supports SSO to customer-owned applications within the Azure AD B2C tenants
Security policy managed by both organizations	Security policy managed by the application
Branding managed by both organizations	Branding managed by the application
Partner users are included in the same Azure AD, by default, with the guest user type	Managed separately from the organization or partner Azure AD

In the next section, we'll compare AD FS against the two services, to provide you with a global view.

Comparing AD FS with Azure B2B and B2C

In this section, we'll provide you with some helpful information so that you can differentiate between AD FS and the Azure AD B2B and B2C functionalities.

We'll start with the main differences between AD FS and the Azure B2B scenario. With Azure AD B2B comes the capability to invite users from partner organizations to access applications on your own Azure AD instance. With AD FS, you can provide the same functionality, with CP trusts, to any partner organization that's based on AD FS.

However, you will hit the following differences:

- With AD FS, you have a lot of flexibility, and can run any customized scenario
- The following requirements need to be fulfilled:
    - Partner requires the Federation Service
    - Certificate handling
    - Administrative overhead

With **Azure AD B2B**, there is an easy invitation process. The feature is available for free, or by using a **5.1 ratio** if basic or premium features are used, and the partner doesn't need to fulfill any requirements—invitations for Azure AD and locally-based directory users, including social identities, are possible.

The admin controls all of the access to your corporate apps through your Azure AD directory. When collaboration is terminated, partner users can be removed from your Azure AD, and their access to your apps will immediately be revoked. Access reviews can be used to manage the life cycles of guest users. Additionally, when a partner user leaves the partner organization, access is automatically lost if the partner organizations uses a native Azure AD and not an AdHoc (unmanaged Azure AD tenant).

Keep in mind that if you still want to use AD FS for B2B, you will need to manage the whole life cycle of the guest user's account.

The next thing that we will discuss is the difference between AD FS and Azure AD B2C. You have two main scenarios that you can run. First, you can handle customer accounts and an on-premise identity store, and the authentication can be managed by AD FS. This scenario is helpful if you want to have full control in your own environment. But you also need to provide several processes and technologies to support such a scenario—and this is for the whole year.

The other option is that customers can sign up or bring their own consumer identities with Azure AD B2C, so you will receive a fully-manageable solution with high availability and the most common processes and support functionalities already implemented.

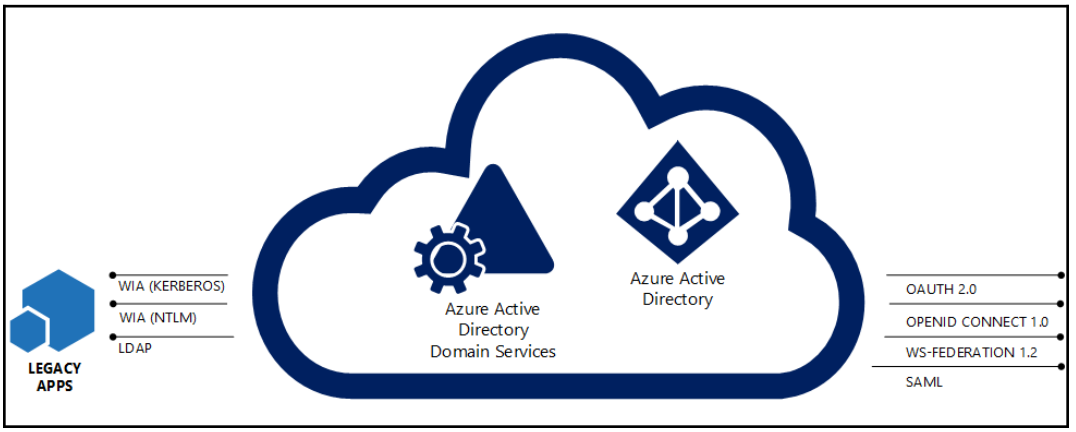
AD FS delivers a very flexible and customizable solution in your own environment, but you have to integrate social media providers and build trust relationships, so that the users can bring their own identities. As we already mentioned, user management and self-servicing have to be built and provided with high availability.

With Azure AD B2C, you receive a fully-packed solution, which allows for the developer of a business application to use the whole identity framework of Azure AD and the B2C extensions. Sign-up pages are ready to use, or users can bring their own identities (Google, Live, Facebook, and so on), and you can enable self-servicing options.

Extending Active Directory solutions with Azure AD Domain Services

Azure AD Domain Services helps you to move your on-premise applications, depending on traditional authentication methods, such as Kerberos and NTLM, to the cloud. This cloud-based service allows you to join your IaaS virtual machines to a managed domain without the need to provide domain controllers on virtual machines. With this solution, you can integrate your applications directly in your Azure Active Directory services and benefit from the rich feature set. With the synchronization of the Azure AD users to Azure AD DS, you can use identities to provide authentication and authorization. You're also able to connect by **Lightweight Directory Access Protocol** (**LDAP/S**) to the directory service.

The following diagram shows the integration scenario, from the perspective of an application installed on an IaaS virtual machine:



Azure AD Domain Services overview

This service provides you with a flat organizational unit structure and group policies, by default, for managing the domain-joined server systems. Generally, it's not a good idea to join client computers in this directory.

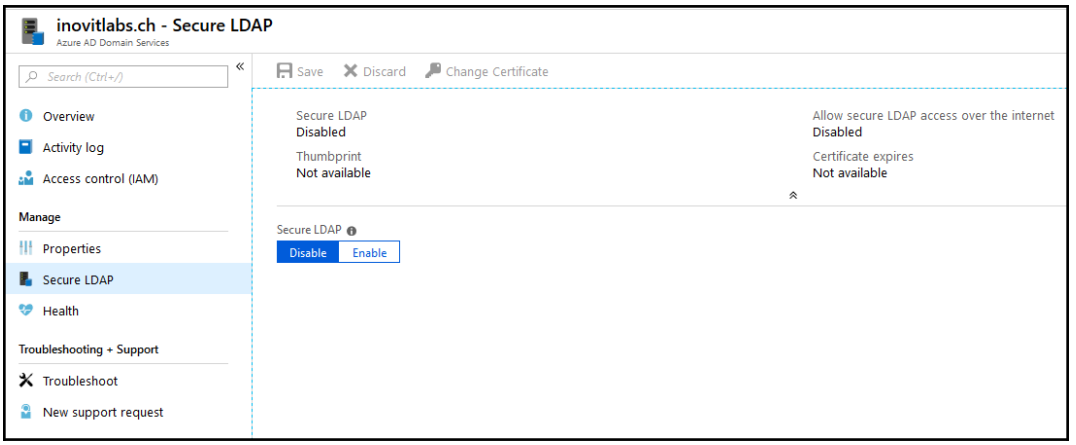
The service is used to help you with the integration of the following systems:

- NAS systems
- Print solutions

It also lifts your on-premise services to Azure if they aren't claims-enabled.

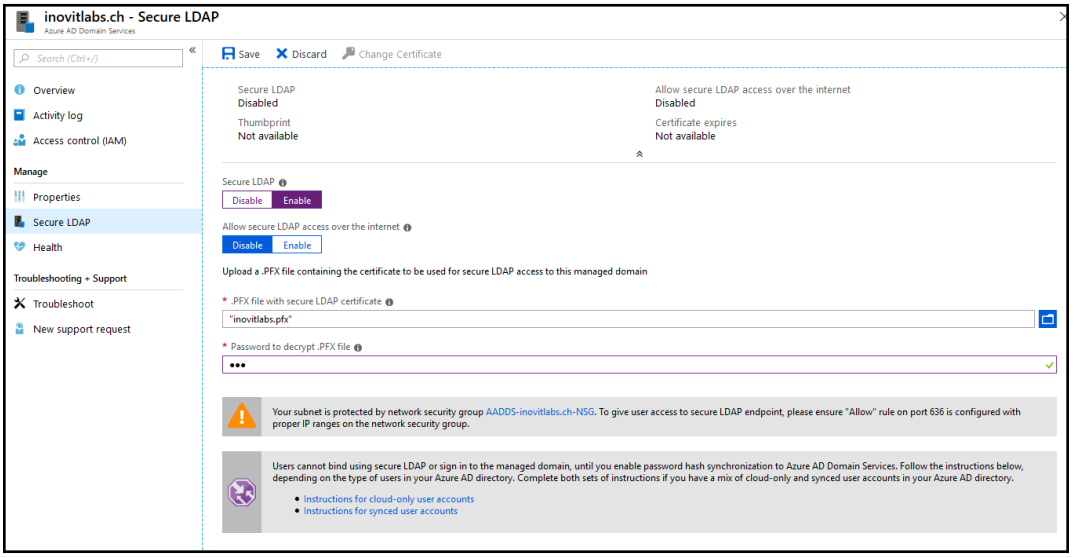
The following configuration shows you how to extend the solution from `Chapter 1`, *Building and Managing Azure Active Directory*, where we first enabled the service. To configure this solution, you will need to have a certificate. Refer to `Chapter 8`, *Using the Azure AD App Proxy and the Web Application Proxy*, to find instructions for creating a certificate for this purpose. Perform the following steps:

1. Navigate to `https://portal.azure.com` on the **Azure AD Domain Services** blade.
2. Choose **Secure LDAP** and **Enable** the service:



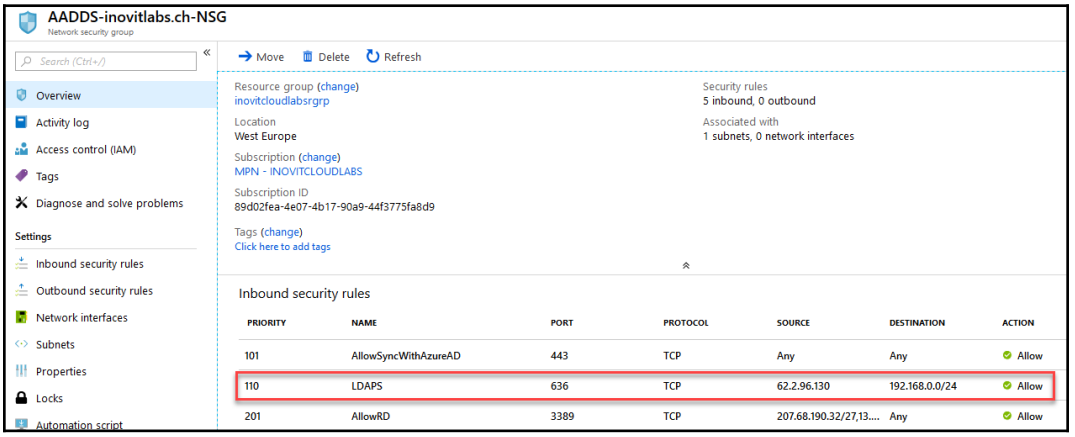
Secure LDAP activation procedure

3. **Enable** the **Allow secure LDAP access over the internet**, and upload the certificate, including the private key:



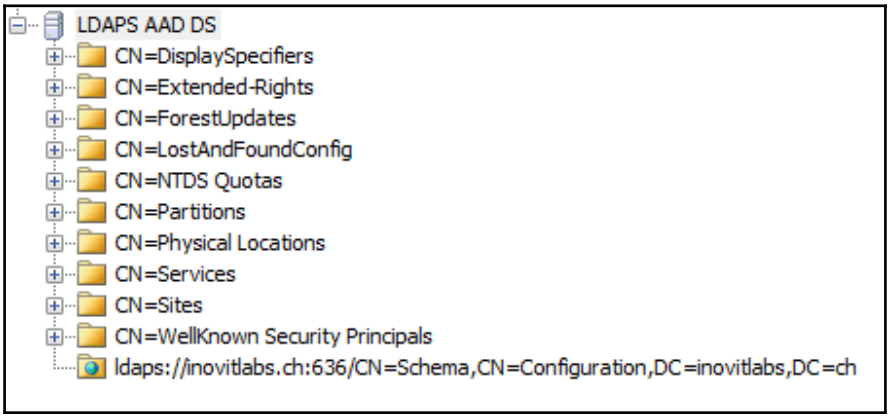
Providing the certificate

4. Create a new LDAPS security rule in the network security group.
5. Add a rule with your source IP and the Azure AD DS destination network:



Changing the firewall rules to allow LDAPS

6. Connect to the LDAPS service with your preferred LDAP browser:



Testing the LDAPS connection

You can find more information about the LDAPS configuration at `https://docs.microsoft.com/en-us/azure/active-directory-domain-services/active-directory-ds-admin-guide-configure-secure-ldap`.

In the next section, we'll discuss how to use AD FS as an on-premise identity service for the cloud.

AD FS as an on-premise identity service for the cloud

Authenticating users in multi-forest environments is just a bit more complex than doing it in a typical single-forest deployment. You should already be aware of the basics of the different authentication protocols and AD FS, thanks to previous chapters. The configuration of the integration with Office 365 is a straightforward process; with the `Convert-MsolDomainToFederated` command, you can create everything that's needed in your AD FS configuration. With the `SupportMultipleDomain` switch, you can define whether you're using a multi-forest scenario.

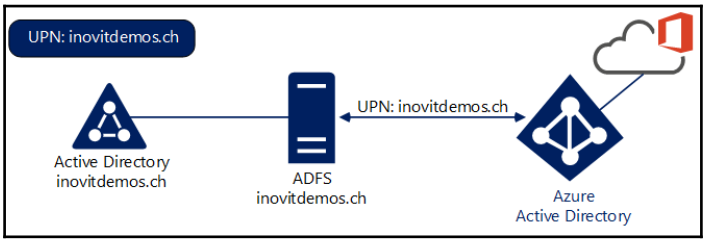
Next, we'll start with the supported and possible scenarios in the case of using multiple forests and Office 365. We'll focus on the AD FS server deployment. Furthermore, you can always attach an AD FS proxy/WAP to these scenarios.

This section will cover the following scenarios:

- A typical single-forest deployment
- Two or more Active Directory forests running separate AD FS instances
- Running one AD FS instance for multiple trusted forests
- Supporting one AD FS instance for multiple Active Directory forests without an AD trust relationship
- Using a local CP trust to support multiple Active Directory forests
- Using a shared Active Directory environment
- Microsoft Cloud Solution Provider summary

Typical single-forest deployment

This scenario is a commonly seen configuration in smaller and medium-sized organizations; it's a one-forest scenario with AD FS authentication to Office 365. You can use one or more UPNs with the related verified UPN domains:

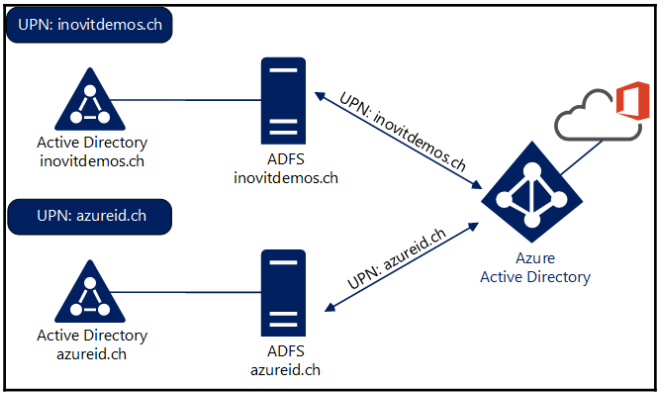


Single-forest deployment

We will follow up with the multiple AD forest scenarios in the next section.

Two or more Active Directory forests running separate AD FS instances

This scenario is commonly used if there are no Active Directory trust relationships and CP trusts are in place. Every Active Directory forest holds its own AD FS server and responds to their own UPN. The administrator only needs to configure a unique UPN suffix. Azure AD Connect will do the relevant identity synchronization for the different forests:

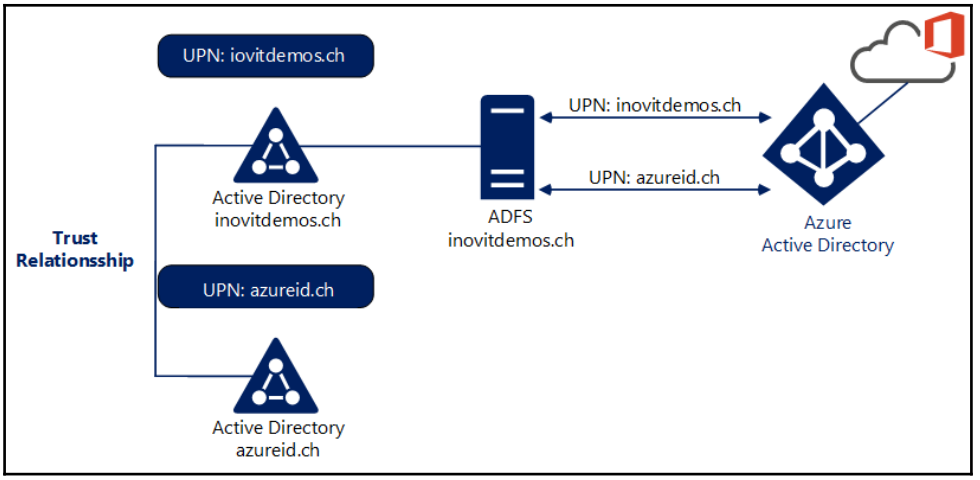


Two or more Active Directory forests running separate AD FS instances

In the next section, we will discuss a one-instance ADFS scenario for multiple forests.

Running one AD FS instance for multiple trusted forests

Many organizations run several Active Directory forests. If a single authentication point is provided, one option is to work with an Active Directory **Trust Relationship**. With this design solution, every forest can use one AD FS environment, and all UPNs are running against this environment:

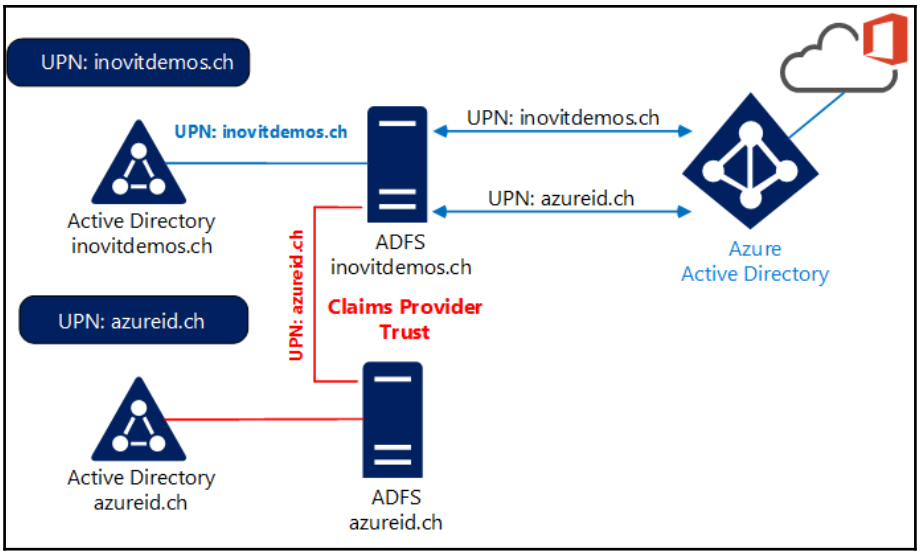


One AD FS instance for multiple trusted forests

Next, we will discuss the one ADFS instance approach for multiple forests without trust relationships.

One AD FS instance for multiple Active Directory forests without an AD trust

Another option for supporting multiple forests is to work with a CP trust, if Active Directory trust relationships can't be used. In this scenario, the AD FS server works, by default, against its own Active Directory forest. The AD FS server will also be configured to ask another AD FS for specific UPNs:

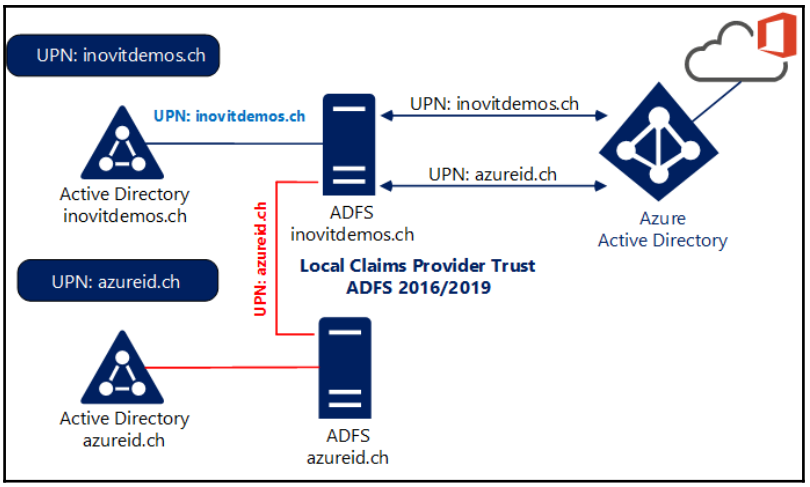


One AD FS instance for multiple Active Directory forests without an AD trust

In the next steps, we will discuss using the local claims provider trust to connect multiple AD forests.

Using a local CP trust to support multiple Active Directory forests

Beginning with AD FS 2016, you have the option of using a local CP trust to integrate additional forests. The only thing that you need to know is that you will lose automatic home-realm discovery for internal users. You can only provide a custom solution to do this for you:



Using a local CP trust to support multiple Active Directory forests

You can configure the scenario with the following procedure:

1. Set the credentials for the service account:

```
$credential = Get-Credential
```

2. Change the `HostName` to your domain controller, and use Port `636` if configured for a secure connection:

```
# Example: $vendorDirectory = New-AdfsLdapServerConnection -
HostName leanads01.leano.ch -Port 636 -SslMode None -
AuthenticationMethod basic -Credential $credential
$vendorDirectory = New-AdfsLdapServerConnection -HostName
yourhostname-Port 636 -SslMode None -AuthenticationMethod
basic -
Credential $credential
$Name = New-AdfsLdapAttributeToClaimMapping -LdapAttribute
sAMAccountName -ClaimType
"http://schemas.microsoft.com/ws/2008/06/identity/claims/windo
wsaccountname"
$Mail = New-AdfsLdapAttributeToClaimMapping -LdapAttribute
mail -ClaimType
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailad
dress"
# Example: $AdditionalAttribute = New-
AdfsLdapAttributeToClaimMapping -LdapAttribute mail -ClaimType
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailad
dress"
```

3. Configure your own identifier, like `urn:example`, and define your root user container, like `DC=EXAMPLE,DC=COM`:

```
# Choose your preferred display name in the Forms-Based
Authentication with the -Name parameter
# Example: Add-AdfsLocalClaimsProviderTrust -Name "Partners" -
Identifier "urn:leano" -Type Ldap -LdapServerConnection
$vendorDirectory -UserObjectClass user -UserContainer
"DC=leano,DC=CH" -LdapAuthenticationMethod basic -
AnchorClaimLdapAttribute //userPrincipalName -AnchorClaimType
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn" -
LdapAttributeToClaimMapping @($Name, $Mail) -
AcceptanceTransformRules "c:[] => issue(claim=c);" -Enabled
$true

Add-AdfsLocalClaimsProviderTrust -Name "External" -Identifier
"urn:dev" -Type Ldap -LdapServerConnection $vendorDirectory -
UserObjectClass user -UserContainer "DC=EXAMPLE,DC=COM" -
LdapAuthenticationMethod basic -AnchorClaimLdapAttribute
userPrincipalName -AnchorClaimType
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn" -
LdapAttributeToClaimMapping @($Name, $Mail) -
AcceptanceTransformRules "c:[] => issue(claim=c);" -Enabled
$true
```

4. Configure additional claim rules:

```
# Build a .txt file with the specific Claim rules - in my case
ruleset.txt
# Change the samAccountName - Domain Suffix to your needs
@RuleName = "Pass through UPN"
c:[Type ==
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn"]
=> issue(claim = c);
@RuleName = "Pass through Mail"
c:[Type ==
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailad
dress"]
=> issue(claim = c);
@RuleName = "Pass through sAMAccountName"
c:[Type ==
"http://schemas.microsoft.com/ws/2008/06/identity/claims/windo
wsaccountname"]
=> issue(Type =
"http://schemas.microsoft.com/ws/2008/06/identity/claims/windo
wsaccountname", Value = "Partners\" + c.Value);
```

5. Set the newly defined claim rules:

```
# Be aware to use the same name as you have chosen under Add-
AdfsLocalClaimsProviderTrust –Name "Partners"
$ruleset = New-AdfsClaimRuleSet –ClaimRuleFile .\ruleset.txt
Set-AdfsLocalClaimsProviderTrust –TargetName "Partners" –
AcceptanceTransformRules $ruleset.ClaimRulesString

Configure the HRD Cookie Lifetime to save the chosen IDP and
to avoid further clicks for the user
# In your case we would recommend a Lifetime of 1825 days = 5
years
Set-AdfsWebConfig –HRDCookieLifetime 90 –
HRDCookieEnabled:$true

Add the additional IDP to your Relying Party Trust
configuration
# TargetName = your RP you want to configure; and the
ClaimsProviderName = Your defined name

Set-AdfsRelyingPartyTrust –TargetName "ClaimsXRay" –
ClaimsProviderName @("Active Directory","Partners")
```

6. Modify the display name of the Active Directory IDP:

```
// Insert the following code to the end of your onload.js
script and define your own display name // Update your custom
theme with the provided Theme Customization Scripts
if ( document.getElementById("hrdArea") ) {
 var strADCPName = "Partners" ;
 //Create an array of all claim provider trust section in the
page
 var listAllSpanForIdp =
document.getElementsByClassName("idpDescription float") ;
 var inc;
 for (inc = 0; inc < listAllSpanForIdp.length; inc++) {
 if ( listAllSpanForIdp[ inc ].innerHTML == "<span
class=\"largeTextNoWrap indentNonCollapsible\">Active
Directory</span>" ) {
 //Change the HTML content of the matching section to the
value specified in the strADCPName variable
 listAllSpanForIdp[ inc ].innerHTML = "<span
class=\"largeTextNoWrap indentNonCollapsible\">"+ strADCPName
+"</span>" ;
 }
 }
}
```

7. Configure the relying party claims issuance policy:

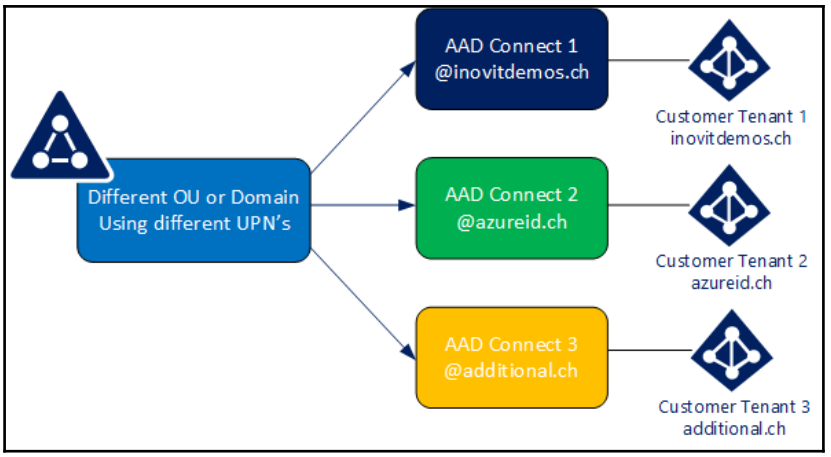
```
# Sends all configured claim definitions from the local AD and
the additional IDP (urn:example)
Claim Issuance Rule 1
c:[Issuer =~ "^(SELF AUTHORITY|LOCAL AUTHORITY|urn:dev)$"]
 => issue(claim = c);
# Removes the domain suffix in the NameID claim
Claim Issuance Rule 2
c:[Type ==
"http://schemas.microsoft.com/ws/2008/06/identity/claims/windo
wsaccountname"]
 => issue(Type =
"http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameide
ntifier", Issuer = c.Issuer, OriginalIssuer =
c.OriginalIssuer, Value = RegExReplace(c.Value, ".+?\\", ""),
ValueType = c.ValueType,
Properties["http://schemas.xmlsoap.org/ws/2005/05/identity/cla
improperties/format"] = "urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified");
# Restart ADFS Service
Restart-service ADFSsrv
```

In the next section, we will discuss using a shared AD environment.

Using a shared Active Directory environment

In recent months, we've had several discussions about using a single forest environment with two Azure Active Directory tenants, including Office 365 services and an AD FS and Web Application Proxy combination. We often get these questions if the organization wants to use a separate Office 365 tenant, operated by 21Vianet or service providers that want to use a shared Active Directory infrastructure for small customers.

The following solution design is based on the following supported AAD Connect topology:



Using a shared Active Directory environment

You can only use this option with AD FS or an equivalent product. Pass-Through Authentication won't work for this use case.

In this topology, an AAD Connect instance is configured for a mutually-exclusive set of objects; for example, an organizational unit or domain. Furthermore, different domains and user principle names need to be used for this scenario, as follows:

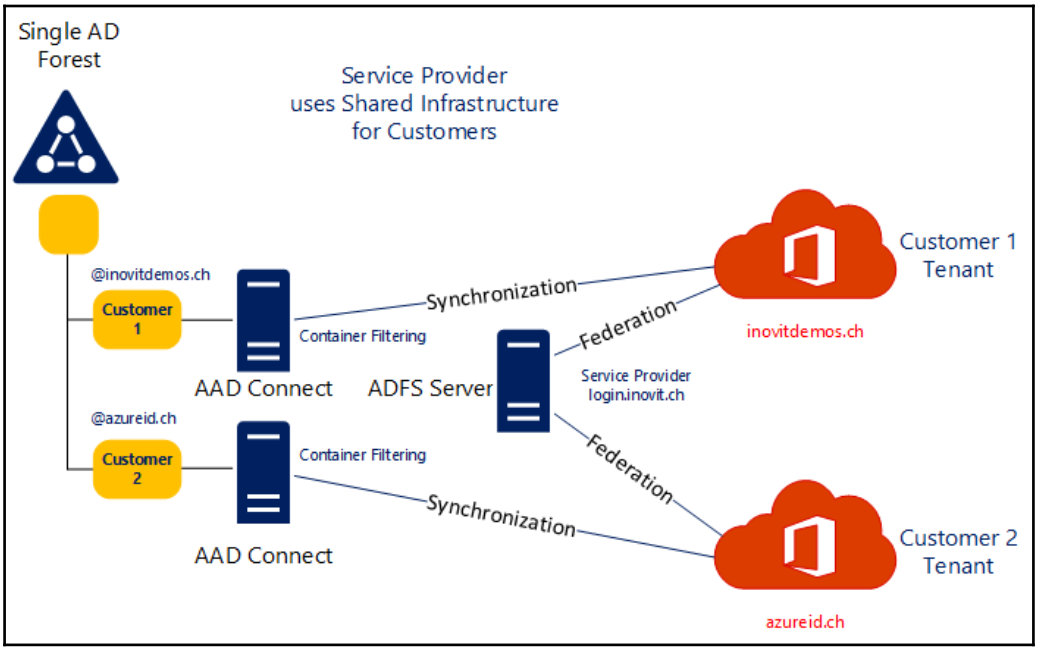
- A DNS name can only be registered in one Azure Active Directory (custom domains)
- One-to-one relationships between an Azure AD Connect synchronization server and one Azure Active Directory
- Azure Active Directory instances are, by design, isolated

The requirement for a mutually-exclusive set of objects also applies to write-back. Some write-back features aren't supported with this topology, including the following:

- Group write-back with a default configuration
- Device write-back

The solution's design is based on the following key features:

- One single Active Directory forest, with organizational units based on regions or customers.
- Users in the organizational units have the associated user principle names configured; for example, **OU APAC** uses the `@apac.inovitdemos.ch` UPN suffix or the **specific customer suffix**, such as `@azureid.ch`.
- Two Azure AD Connect instances configured with the container filter, based on the organizational units.
- One AD FS and Web Application Proxy combination, with the `login.inovit.ch` STS name.
- One Azure Active Directory Tenant with Office 365 services, called **Customer Tenant 1**, and the following registered custom domain name: `inovitdemos.ch`.
- One Azure Active Directory Tenant with Office 365 services, called **Customer Tenant 2**, and the `apac.inovitdemos.ch` or `azureid.ch` registered custom domain name:



Federation configuration overview

The following description provides the main configuration steps to implement this scenario in your environment:

- **Configuration of the Federation Trust for Customer Tenant 1**: This task is a very common one, because you just need to open an evaluated PowerShell on the AD FS server and type the following commands:

```
Connect-MsolService #Enter your global administrator
credentials
Convert-MsolDomainToFederated -DomainName inovit.ch -
SupportMultipleDomain
```

- **Configuration of the Federation Trust for Customer Tenant 2**: You can't use the same PowerShell command for the second tenant's configuration. If you try, you'll change the configuration completely to the second tenant. In this case, we need to use the `Set-MsolDomainAuthentication` command to configure the trust to the second tenant.

`Set-MsolDomainAuthentication` is typically used to configure Federation Trusts with other Identity Providers.

To configure the second Federation Trust, you need to export the AD FS token-signing certificate from the AD FS farm configuration. You can do this with the AD FS management console or the following PowerShell commands:

```
$certTS=Get-AdfsCertificate -CertificateType Token-Signing
$certInf=$certTS[0].Certificate.Export([System.Security.Cryptography.X
509Certificates.X509ContentType]::Cert)
[System.IO.File]::WriteAllBytes("c:\temp\inovit-ts.cer", $certBytes)
```

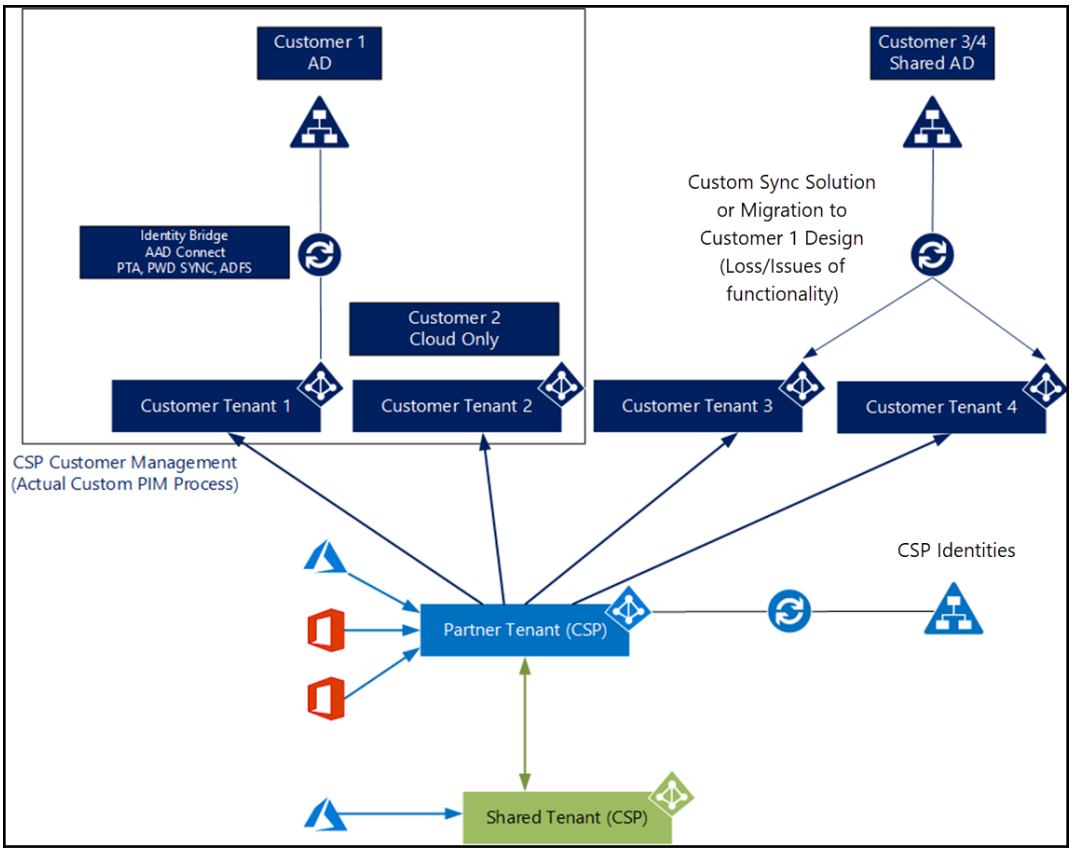
Now that we have exported the token-signing certificate, we can start to configure the Federation Trust for **Customer Tenant 2**:

```
$crt = New-Object
System.Security.Cryptography.X509Certificates.X509Certificate2("c:\tem
p\inovit-ts.cer ")
$certData = [system.convert]::tobase64string($cert.rawdata)
$customdomain="azureid.ch"
#customdomain="apac.inovitdemos.ch"
$url="https://login.inovitdemos.ch/adfs/ls/"
$uri="http://login.inovitdemos.ch /adfs/services/trust/"
$ura="https://login.inovitdemos.ch/adfs/services/trust/2005/usernamemi
xed"
$logouturl="https://login.inovitdemos.ch/adfs/ls/"
$metadata="https://login.inovitdemos.ch/adfs/services/trust/mex"

Set-MsolDomainAuthentication -DomainName $customdomain -Authentication
Federated -ActiveLogOnUri $ura -PassiveLogOnUri $url -
MetadataExchangeUri $metadata -SigningCertificate $certData -IssuerUri
$uri -LogOffUri $logouturl -PreferredAuthenticationProtocol WsFed
```

With this solution, you're also ready to solve other scenarios with the same requirements.

Microsoft Cloud Solution Provider summary

The following figure provides an overview of the best way to integrate and connect cloud services for Microsoft Cloud Service Providers. The most supported way is to use one Active Directory for the customer and connect a single Azure AD tenant, or move a small customer to a cloud-only scenario. Azure AD DS can be very helpful in such a scenario:



Microsoft Cloud Solution Provider overview

The shared local Active Directory scenario can still be used, but with great effort.

Summary

In this chapter, you worked through the different Azure AD identity services, such as Azure AD B2B/B2C and the Domain Services. You also explored different options with AD FS as an identity service, in order to get the whole picture of a hybrid identity and the access-management world.

In the next chapter, we'll dive into different application types and deployment methods, including more details about conditional access and other features that we can use.

11
Creating Identity Life Cycle Management in Azure

Handling the identity life cycle in cloud services is a highly requested topic; in particular, we are concerned about the handling of guest users in the Azure Active Directory, providing access to applications in the cloud and on-premise, including the sharing of information in a typical collaboration scenario. Precisely for this reason, we will discuss many use cases for handling guest users securely in your environment, and we will provide good usability for the users to access applications and data. We will also consider some helpful tools and services to automate your identity life cycle management. For sure, we can only provide a small footprint of ideas and references in this chapter, because there are many tasks and ideas to solve.

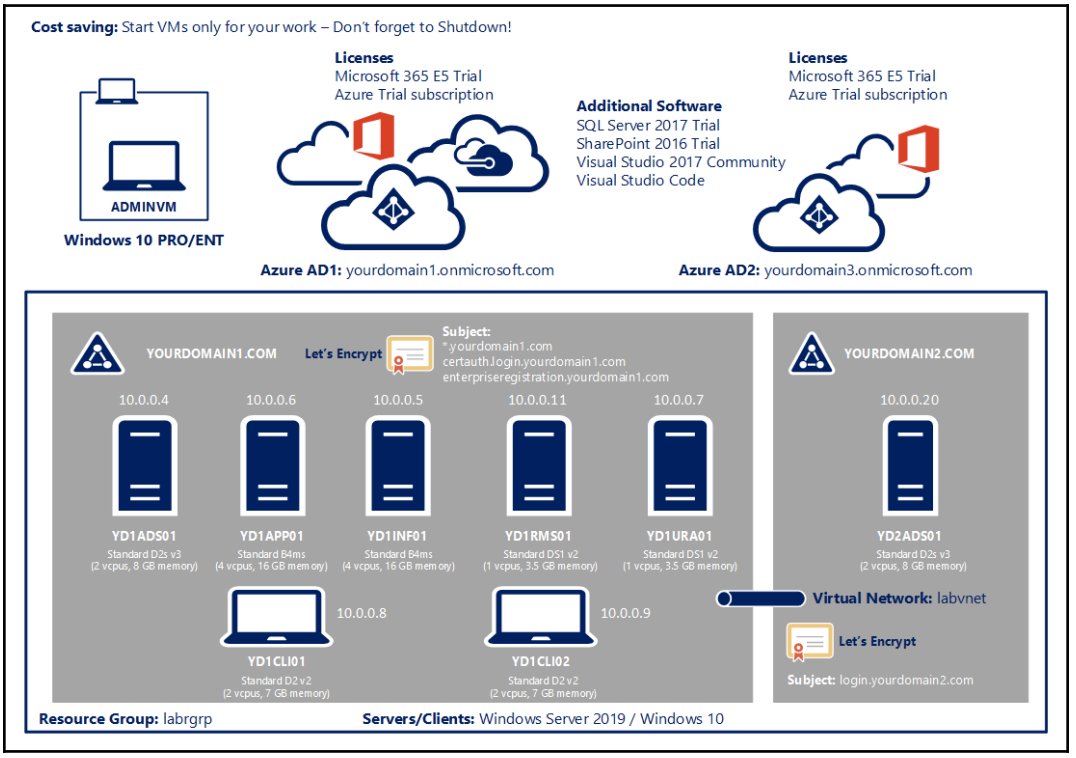
The chapter will be organized into the following sections:

- Lab environment readiness
- Handling the guest user life cycle
- Azure services for automation

We will be working practically on different topics, so we will need to prepare some prerequisites in our test environment.

Lab environment readiness

For this chapter, we will need a configured second Azure AD with some test users inside. The users need to be licensed with Office 365 E3 or E5 licenses. The tenant and the associated public DNS configuration for the additional custom domain need to be done so that the users can send and receive emails. `Chapter 1`, *Building and Managing Azure Active Directory*, provided you with the required technical references to be ready to use the lab configuration in this chapter. Testing the functionality with the Azure AD application proxy requires that you have finished the steps in `Chapter 1`, *Building and Managing Azure Active Directory*, or in `Chapter 9`, *Deploying Additional Applications on Azure AD*; specifically, the Kerberos application publishing. We will do the configuration to provide external access to on-premise applications for guest users on the **YD1APP01** virtual machine, as described in the following diagram:

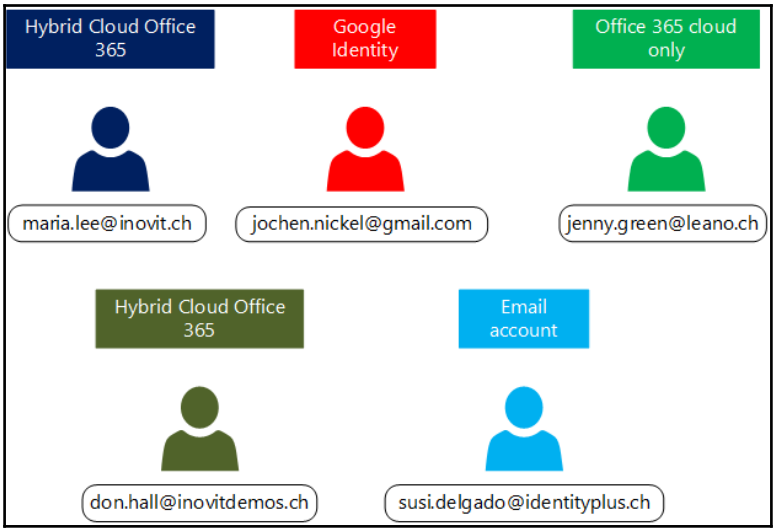


Lab environment overview

In addition to the infrastructure requirements, we will introduce you to the test user sets for the different invitation flows, to get a better understanding of all of the related identity life cycle tasks. To be clear which user is based on which infrastructure, we have put the following small explanations in place:

- **Marie Lee**: Create in a hybrid cloud Office 365 partner infrastructure/tenant (`inovit.ch`)
- **Jochen Nickel**: Test user in Google; if you don't have one, create one for your tests (Google)
- **Jenny Green**: Create in a cloud-only Office 365 partner environment (`yourdomain3.onmicrosoft.com/leano.ch`)
- **Don Hall**: Internal user and inviter of the `yourdomain1.onmicrosoft.com` tenant (`inovitdemos.ch`)
- **Susi Delgado**: Create an external email account (`identityplus.ch`) without Azure AD integration

This is summarized in the following diagram:



Test users for the different Azure B2B scenarios

Now that we have prepared our infrastructure and our test user setup, we can start to handle the guest user life cycle.

Handling the guest user life cycle

In the following section, we will work through the different identity life cycle tasks for guest users. We will organize this section into specific use cases, as follows:

- **Use Case 1**: Exploring the invitation process with different user types
- **Use Case 2**: Using the Azure AD B2B portal
- **Use Case 3**: Providing guest user access to on-premise apps

Now, we will start with the first use case.

Use Case 1 – Exploring the invitation process with different user types

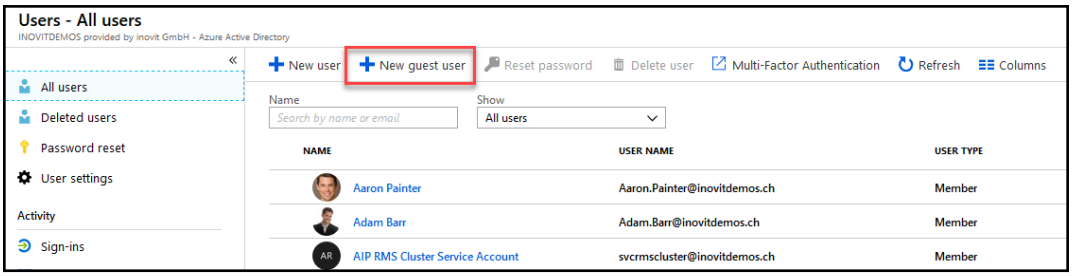
In the following use case, we will explore the invitation process for different user types. We will use our global administrator to invite the following users:

- Maria Lee
- Jochen Nickel
- Jenny Green
- Susi Delgado

The usernames need to be replaced with your names.

With the next steps, we will start the configuration:

1. Open the Azure portal, `https://portal.azure.com`, as the global administrator, and navigate to the Azure AD blade.
2. Navigate to **Users** | **All users**, as follows:



Guest user creation process in Azure AD portal

3. Click on **New guest user** and invite all of the four test guest users, and provide a short personal message, like `Welcome to the team!`
4. You can also use a bulk import over a CSV file.
5. The basic structure of the file needs to be as follows:

Name	Invited User EmailAddress
Maria Lee	maria.lee@inovit.ch
Jochen Nickel	jochen.nickel@gmail.com
Jenny Green	jenny.green@leano.ch
Susi Delgado	susi.delgado@identityplus.ch

6. Afterward, you can run the following commands.
7. Connect to your Azure AD tenant, as follows:

```
Connect-AzureAD -TenantDomain "YOURDOMAIN1.onmicrosoft.com"
```

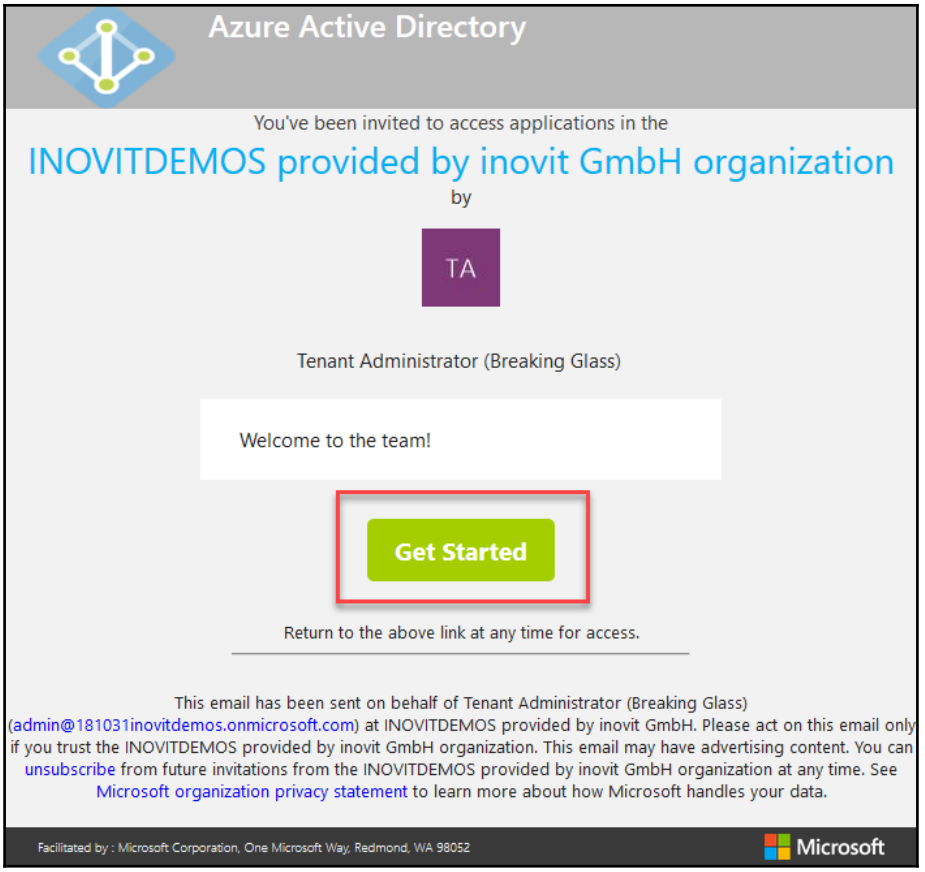
8. Execute the invitation, as follows:

```
$invitations = import-csv <Path to your invitation CSV file>
$messageInfo = New-Object
Microsoft.Open.MSGraph.Model.InvitedUserMessageInfo
$messageInfo.customizedMessageBody = "Welcome to the team!"
foreach ($email in $invitations)
 {New-AzureADMSInvitation `
 -InvitedUserEmailAddress $email.InvitedUserEmailAddress `
 -InvitedUserDisplayName $email.Name `
 -InviteRedirectUrl https://myapps.azure.com `
 -InvitedUserMessageInfo $messageInfo `
 -SendInvitationMessage $true
 }
```

9. Check the newly created guest users, as follows:

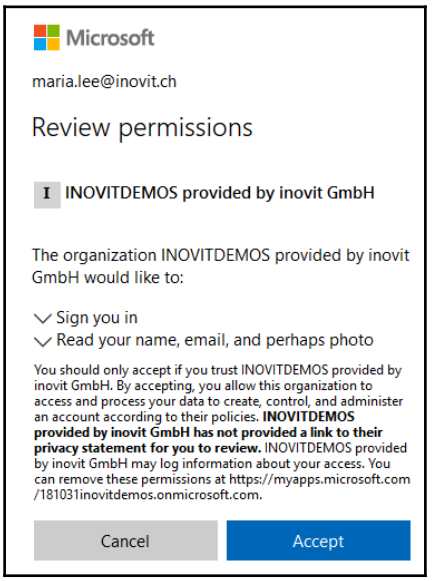
```
Get-AzureADUser -Filter "UserType eq 'Guest'"
```

10. Next, we will accept all of the invitations, in the following order:
    - Marie Lee
    - Jenny Green
    - Jochen Nickel
    - Susi Delgado
11. Log in to every mailbox to accept the invitation (use a different browser or InPrivate sessions).
12. You should receive a message like this on every account:



Azure AD guest invitation message

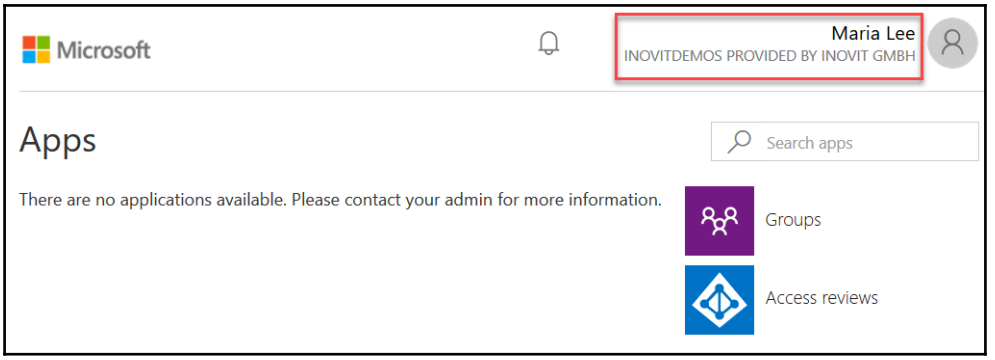
13. Click on **Get Started** as Maria, and you will be redirected to the consent of `YOURDOMAIN1.ONMICROSOFT.COM`, as follows:



Permission consent

This is the default behavior for Azure AD based users, and will be the same with **Jenny Green**.

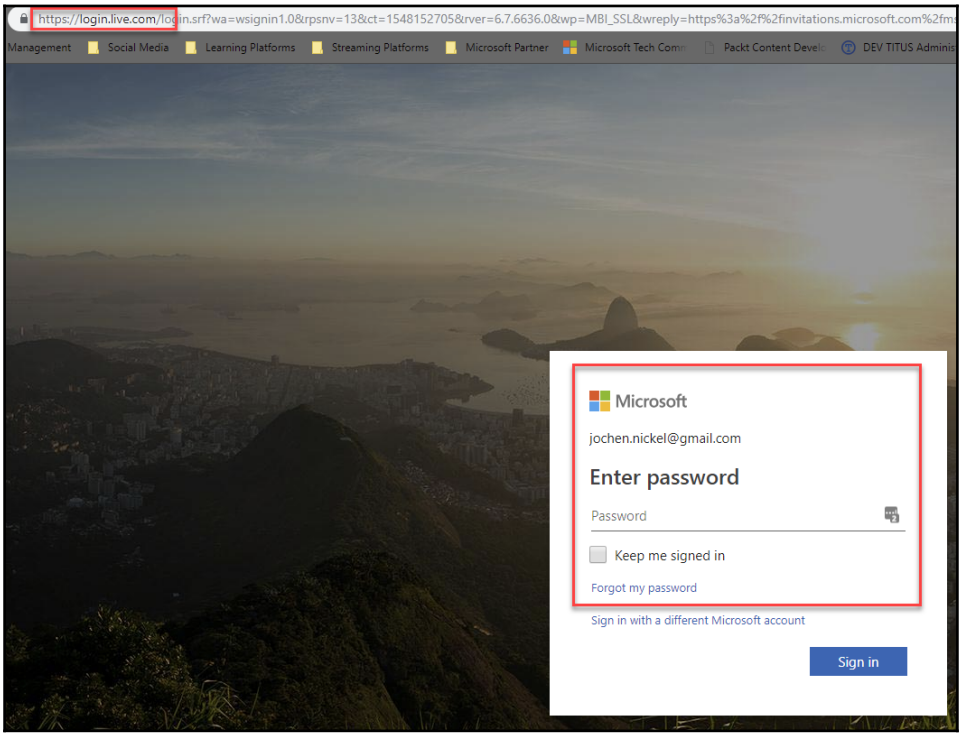
14. Accept the permissions, and you will automatically be signed in to the access panel UI of `YOURDOMAIN1`, as follows:



Azure AD Access Panel UI

With this process, you will have no preassigned applications and groups.

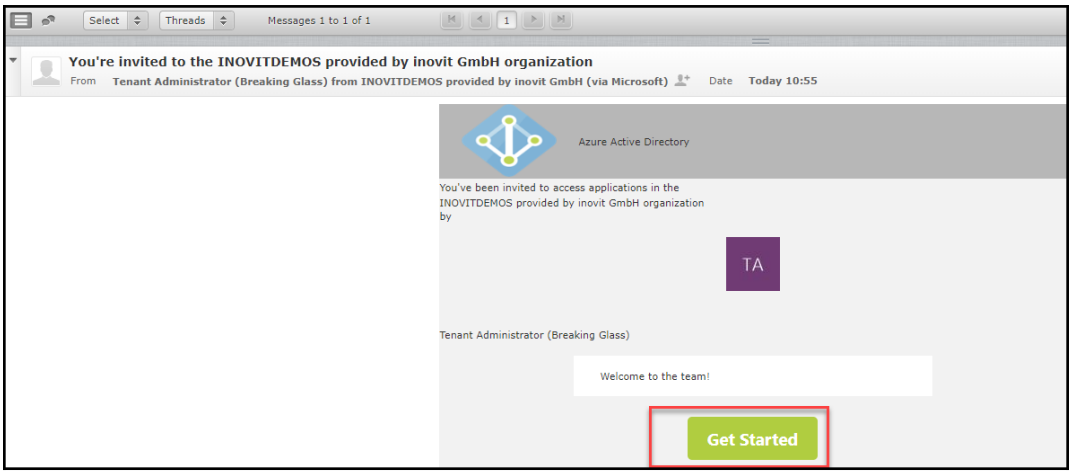
15. Log out and do the same for Jenny Green, and you will have the same behavior experience.
16. Next, you will do the same with your Gmail account; in my case, it's `jochen.nickel@gmail.com`.
17. You will notice that you are redirected to `https://login.live.com`, and you need to log in with your Gmail account credentials, as follows:



Login experience of the Gmail account to live.com

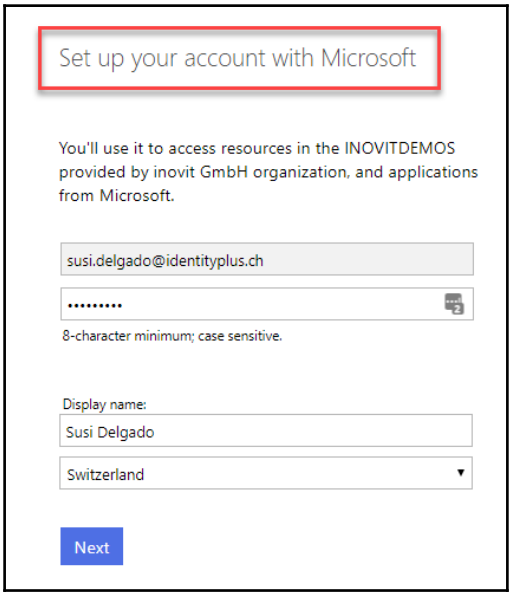
This is the typical behavior for Gmail and other federated identity providers to Azure AD.

18. The other steps are the same as for existing Azure AD users.
19. Log out and run the process for the external non-Azure AD email account (in my case, Susi Delgado) as follows:



Email account behavior for guest invites

20. You will notice that you need to create a Microsoft account, as follows:

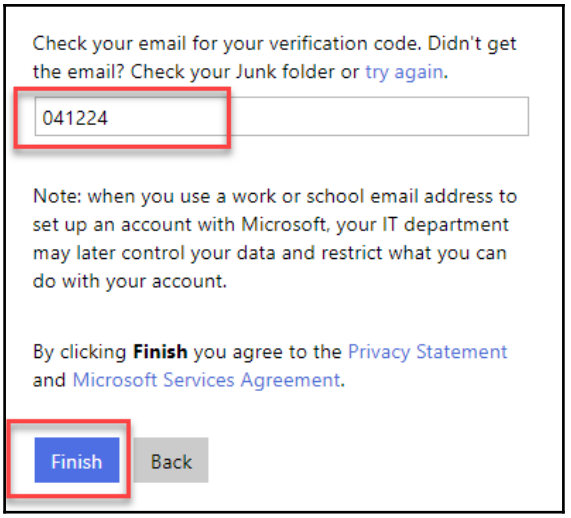


Microsoft account setup

This is the default behavior if the user doesn't exist in any Azure AD.

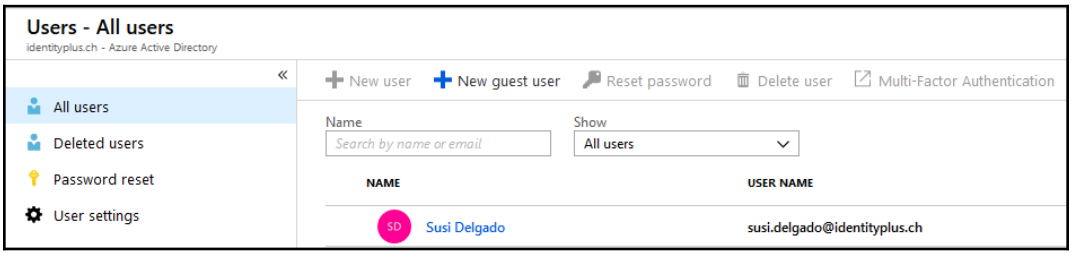
You will see the following link in the address bar: `https://invitations.microsoft.com/signup?tenant=7709ca2b-3be8-4d92-89d7-dc1e274b4d0e`

21. You will receive a verification email to your account.
22. Enter the **verification code** and click on **Finish**, as follows:



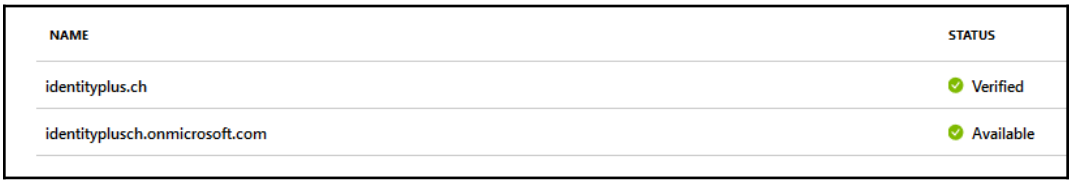
Verification process

23. Accept the consent that appears. Technically, you are now the owner of your own Azure AD tenant.
24. You can verify this by opening the Azure portal, `https://portal.azure.com`, and logging on as **Susi Delgado**.
25. Navigate to the Azure AD blade and check the existing users, as follows:



Azure AD user management

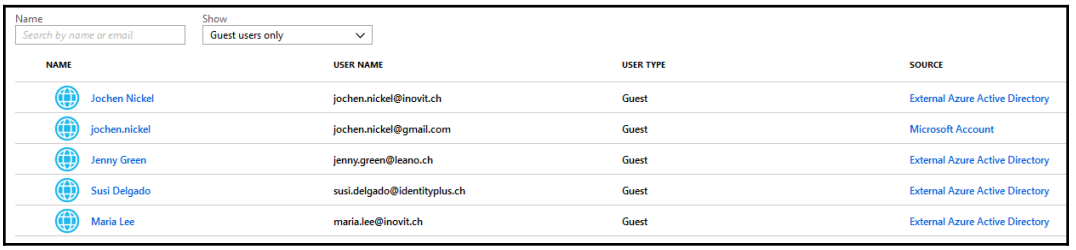
26. Check the custom domains, and you will find that your suffix is mentioned as **Verified**, as follows:



Custom domain overview

With the following procedure, you can take over the Azure AD tenant `https://docs.microsoft.com/en-us/azure/active-directory/users-groups-roles/domains-admin-takeover`.

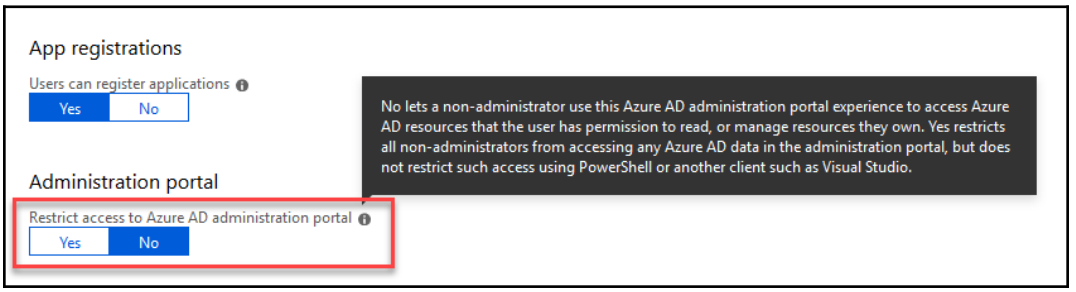
You will find all guest users with different source types, as shown in the following screenshot:



Azure AD user management—Guest users only filter

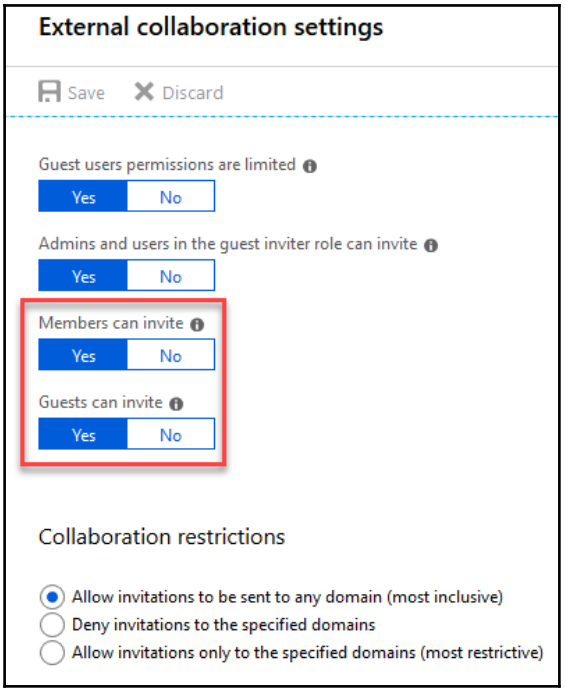
To summarize the invitation process up to now: if a user has already managed in a partner Azure AD, there is no special behavior, except for accepting the invitation and the permission consent. A user from Gmail, or any other federated identity provider, will be treated on a special `live.com` directory. Furthermore, a user that doesn't already exist on any Azure AD will be created in an unmanaged Azure AD. Be aware that there is no automatic deprovisioning process, and the user will have to use an additional credential set.

With the following configuration switch, you can limit users to accessing the Azure portal. If you run an Azure AD Privileged Identity Management scenario, the following will be not the best option:



Option to restrict Azure AD Portal access

Furthermore, you should reconfigure the default settings for the external collaboration so that guests can't invite other guests, as follows:



Azure AD guest user handling options

Another useful tip is how to convert a guest user to an internal member. This can be reached with the following cmdlets:

1. Provide global administrator credentials, as follows:

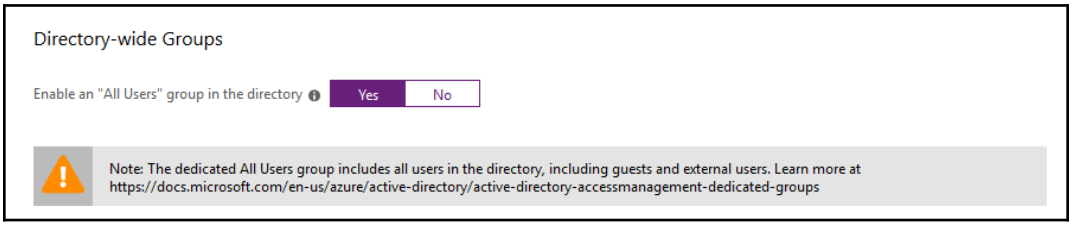
```
Connect-AzureAD
```

2. Convert the user, as follows:

```
Get-AzureADUser -SearchString
UserPrincipalName@YOURDOMAIN1.COM | Set-AzureADUser -
UserType member
```

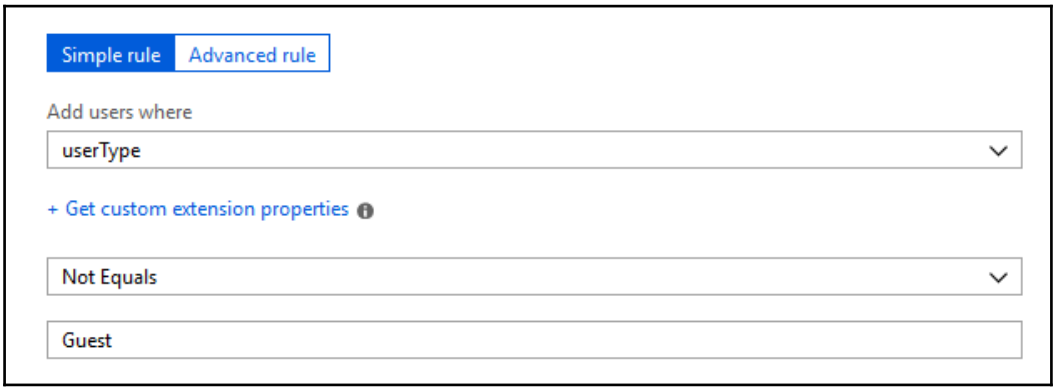
The next important point is to harden the **All users** group, if enabled. To do this, take the following steps:

1. You will find the setting under the Azure AD blade | **Groups** | **General**, as shown in the following screenshot:



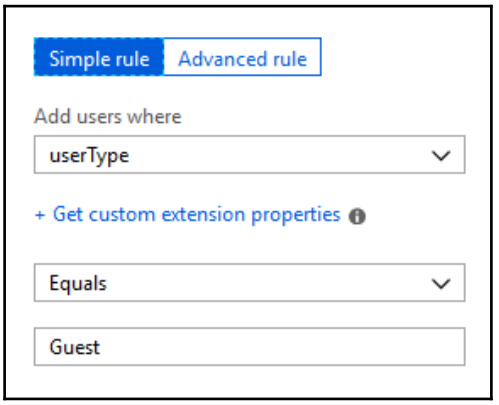
All users option

2. Do the following hardening configuration on the **All users** group:



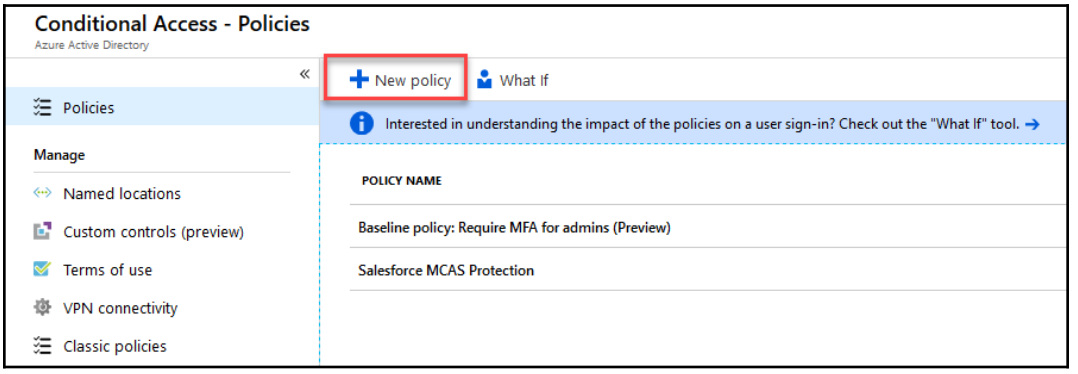
Dynamic group membership rule for the member user type

3. Build another group with dynamic group membership, like **All Guest Users**, as follows:



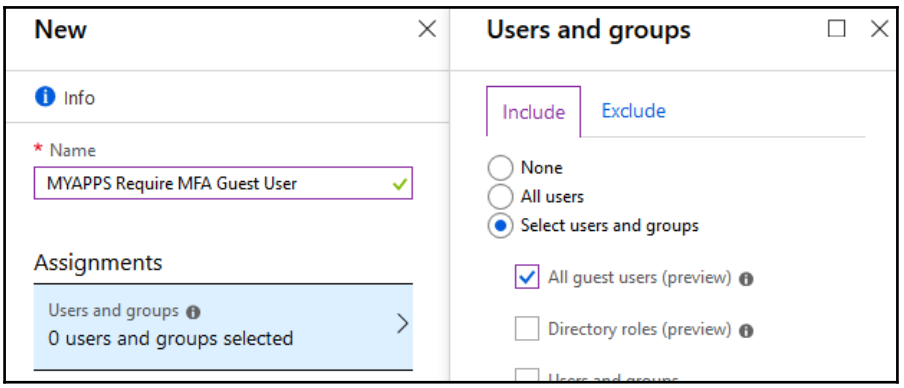
Dynamic Group membership rule for guest users

4. You can also enable Azure **multi-factor authentication** (**MFA**) for all guest users to access the Azure portal, `https://portal.microsoft.com`.
5. Navigate to **Conditional Access** (Azure AD Premium P2).
6. Click on **New policy**, as follows:



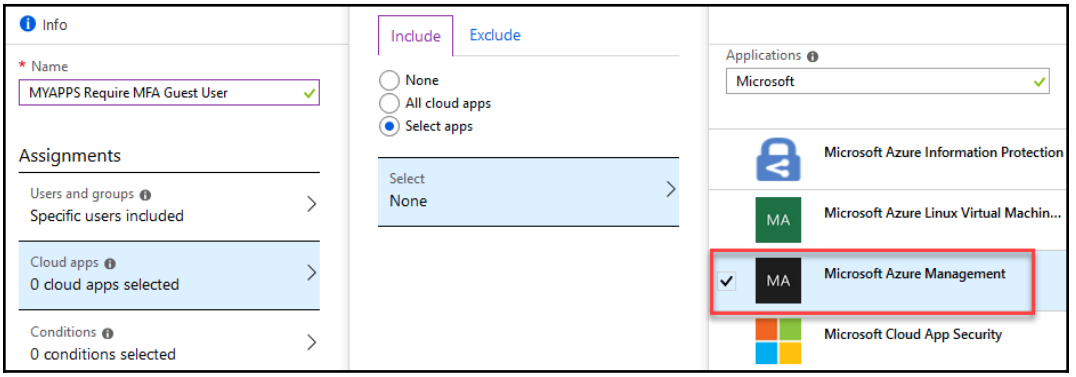
Conditional Access management policies for Salesforce

7. Define a name and choose **All guest users**, as follows:



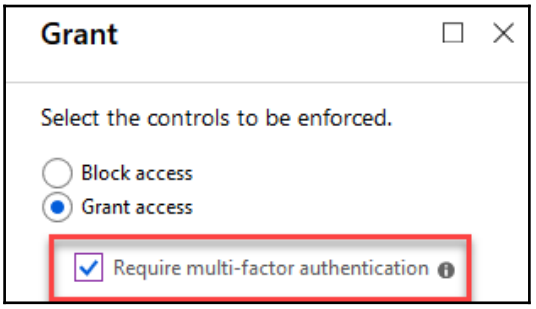
Guest user selection

8. Choose the **Microsoft Azure Management** application, as follows:



Protection of the Azure Portal access

9. Under **Access Control**, select **Require multi-factor authentication**, as follows:



MFA requirement

10. Enable the policy.
11. Test the functionality with one of your guest users.

After working through the first processes, we can extend our solution with the Azure AD B2B portal, in order to provide more capabilities.

Using the Azure AD B2B portal and use cases

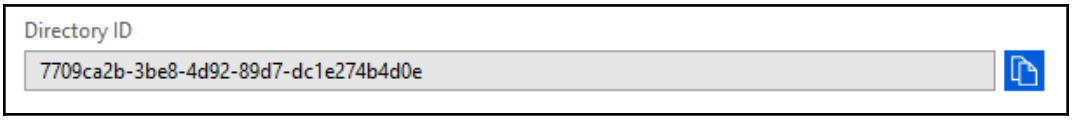
The invitation API provides a lot of capabilities to customize the onboarding workflow for guest users. You can include processes like self-signup, assigning a manager for guest users, an update to the user profile, and an automatic group or application assignment to the guest users. We will use the Microsoft sample project to demonstrate the different capabilities in your environment. You can find the sample project at `https://github.com/Azure/active-directory-dotnet-graphapi-b2bportal-web`.

Installation and configuration

In the beginning, we need to provide two Azure Active Directory applications that will be used to delegate the correct permissions to the Azure B2B sample portal. To do this, take the following steps:

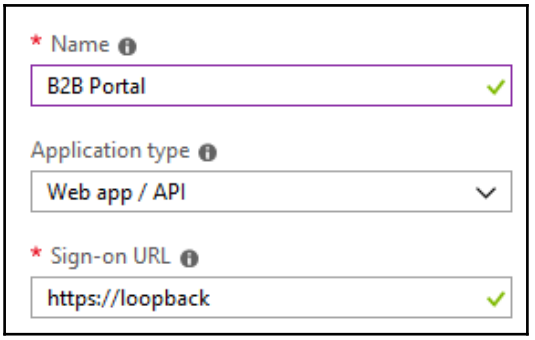
1. Navigate to your Azure portal, `https://portal.azure.com`, and the Azure AD blade.

2. Click on **Properties** and copy the directory ID to a notepad, as follows:



Getting directory ID

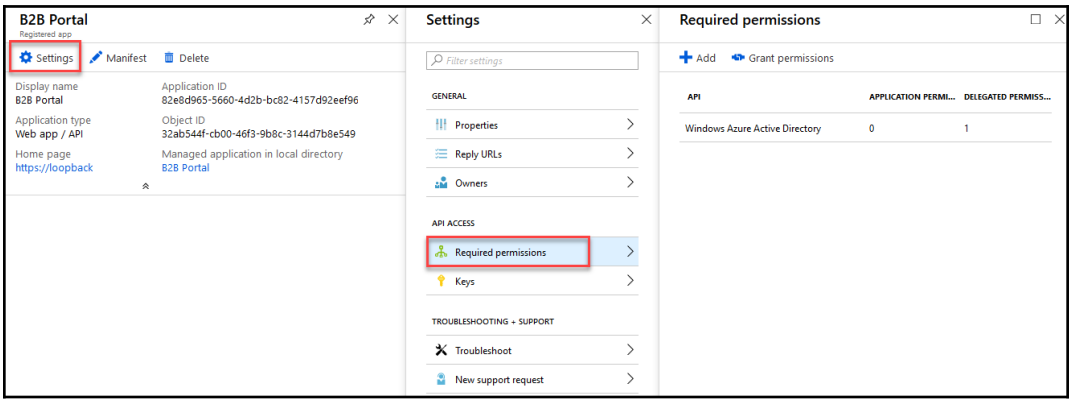
3. Click on **App registrations** and add a new app, like the following:



Configuring app properties

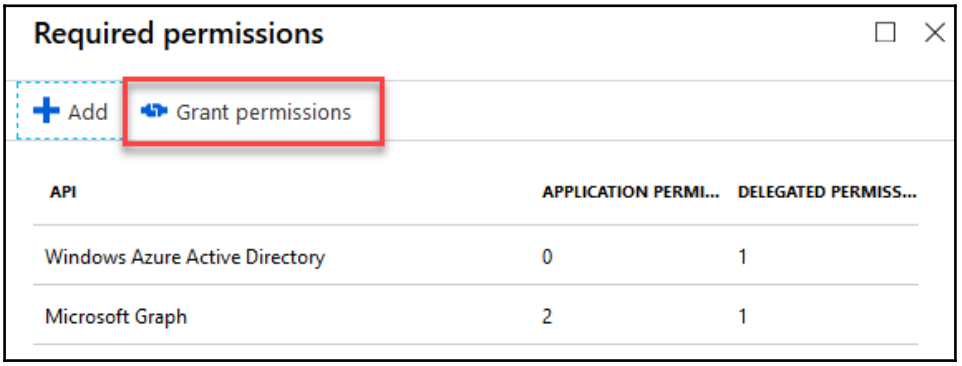
4. Click on **Settings** on the newly created app.
5. Navigate to **Required permissions**.
6. Click on **Add** and select the **Microsoft Graph**.
7. Click on **Select permissions** and enable the following permissions:
    - Application permissions:
        - Read and write directory data
        - Read and write all users' full profiles

- Delegated permissions:
    - Sign in and read user profile, as follows:



Configuring required permissions

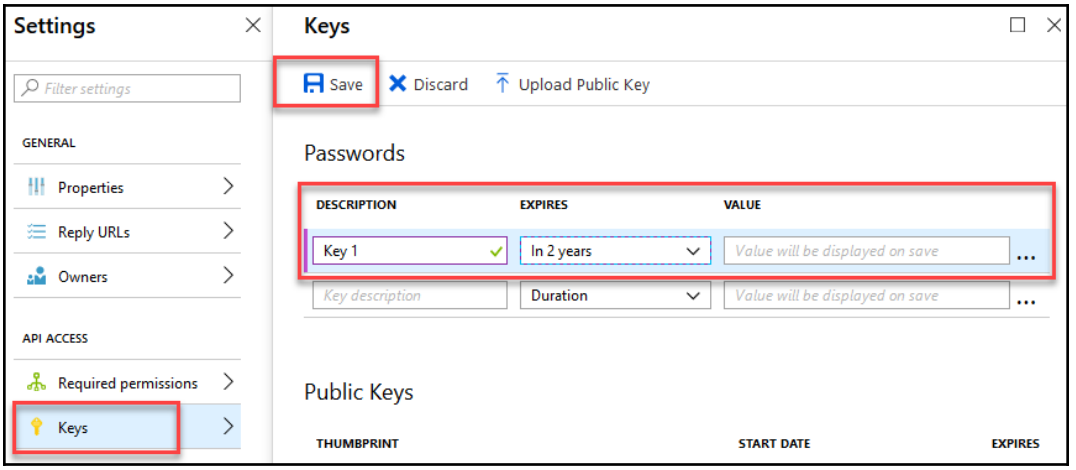
8. After you have added the **Microsoft Graph** API and configured the permissions, click on **Grant permissions**, as follows:



Permission granting procedure

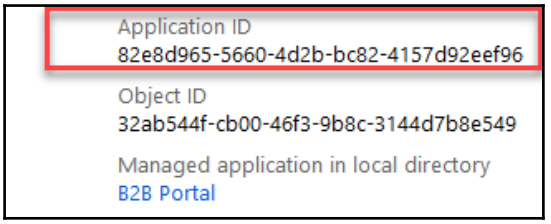
9. Navigate to the **Keys** section.
10. Provide a key description and choose expires in 2 years.

11. Click on **Save** and copy the key value to your notepad, as follows:



Key generation

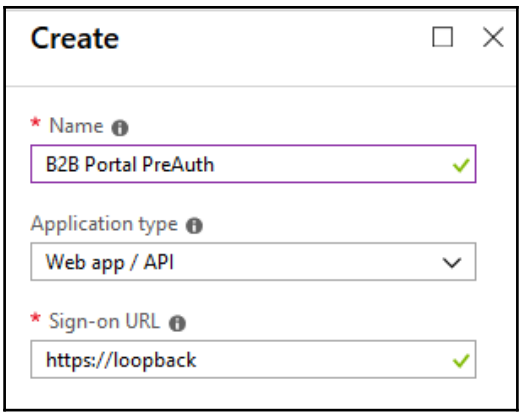
12. Next, copy the **Application ID** to your notepad, as follows:



App ID gathering

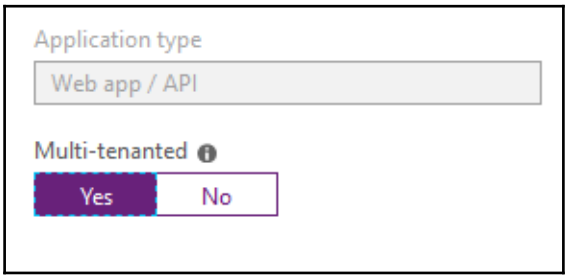
13. Now, we can create the second app for the pre-authentication.

14. Use the following values:



Pre-auth app configuration

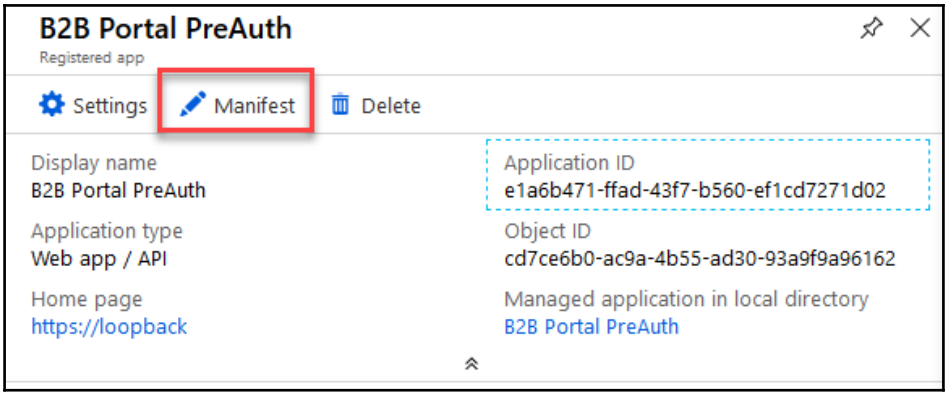
15. Under **Settings** | **Properties**, choose **Multi-tenanted** as **Yes**, as follows:



Multi-tenant option enabled

16. Click on **Required permissions** and add the **Microsoft Graph** API.
17. Assign the following permissions:
    - **Delegated permissions**: Sign in and read user profile
18. Don't forget to press the **Grant permissions** button.
19. Next, we need to generate an app key, like we did for the first app.
20. Copy the key value to your notepad.
21. Copy the **Application ID** to your notepad.

22. Click on **Manifest**, as follows:



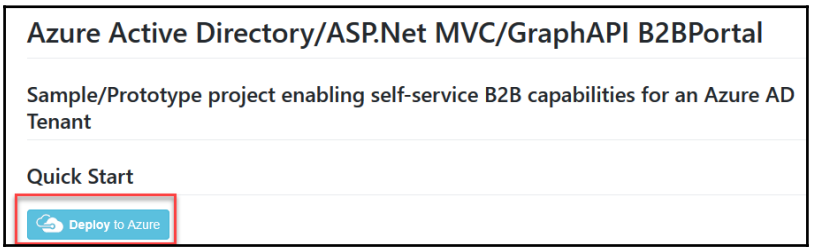
App manifest configuration options for advanced topics

23. We need to change the **oauth2AllowImplicitFlow** value to `true`, as follows:

```
"keyCredentials": [],
"knownClientApplications": [],
"logoutUrl": null,
"oauth2AllowImplicitFlow": true,
"oauth2AllowUrlPathMatching": false,
"oauth2Permissions": [
```

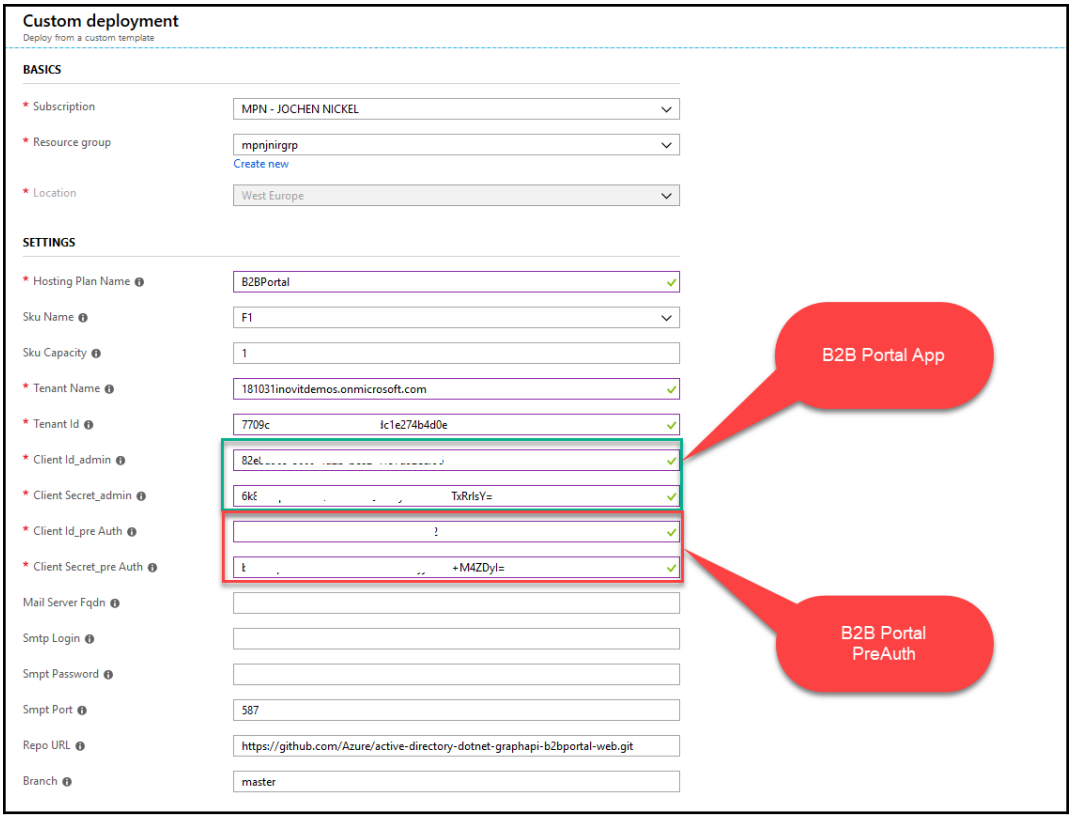
OAuth flow option

24. Now, we can start to deploy our web application to Azure.
25. Click on **Deploy to Azure**, as follows:



Example of portal deployment to Azure

26. Use the values from your notepad and fill, them in as follows:



Azure AD B2B portal deployment options

27. Wait until the deployment process is done.
28. Navigate to the **App Service** section on the Azure portal.

29. Copy the URL to your **B2B Portal** application, as follows:

App deployment result

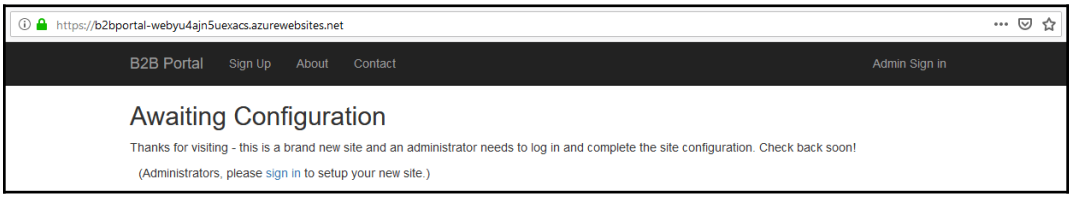
30. Edit each of your Azure AD apps and change the Homepage and Reply URL.

All done. Now we can start to use the portal and build some example processes.

Usage of the portal

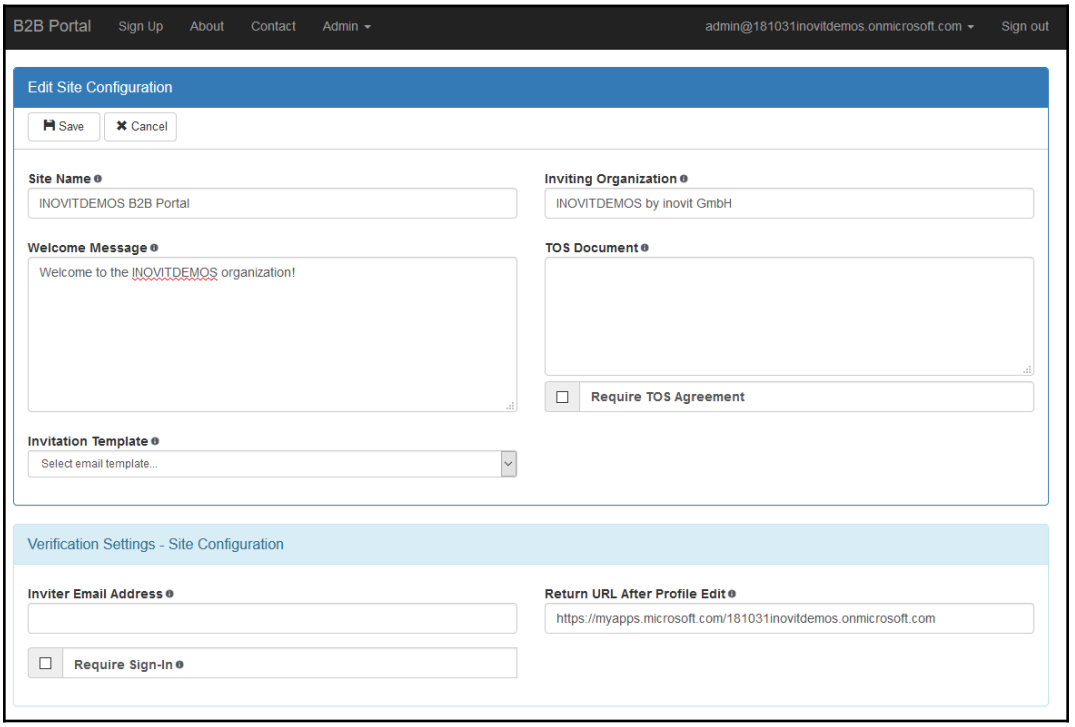
The Azure AD B2B portal provides many processes to show the power of the invitation API. With the first touch on your brand new site, you need to log in and configure the basic parameters. To do this, take the following steps:

1. Open your portal link and log in to the Azure AD B2B portal with your global administrator credentials, as follows:



Azure AD B2B Portal basic configuration

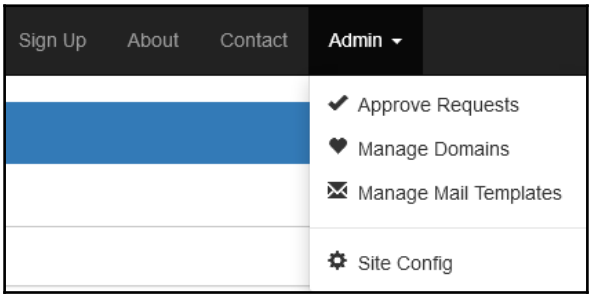
2. In the next configuration steps, you can configure the corporate identity of the web page, including the communication templates, as follows:



Azure AD B2B example portal site configuration

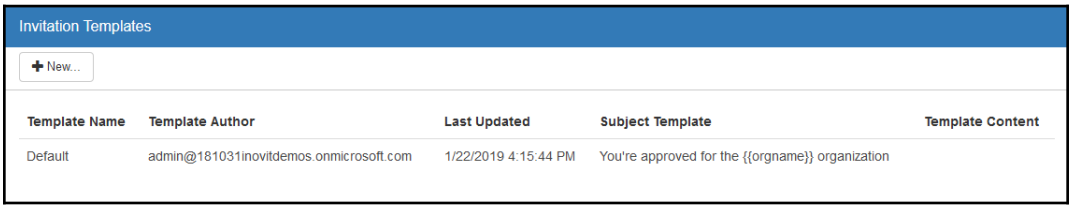
3. Enter the site name and the inviting organization, including a short welcome message.
4. Enter the inviter email address and click on **Save**.
5. Next, we will create three more user accounts in your partner Azure AD tenant, `yourdomain3.onmicrosoft.com` (`leano.ch`).

6. Assign an **Office 365 E3/E5** license to provide a mailbox for the users.
7. Just name them as follows:
    - `guest.user1@leano.ch`
    - `guest.user2@leano.ch`
    - `guest.user3@leano.ch`
8. Now, we can see the capability of providing custom email templates for the invitation process.
9. Click on **Admin** | **Manage Mail Templates**, as follows:



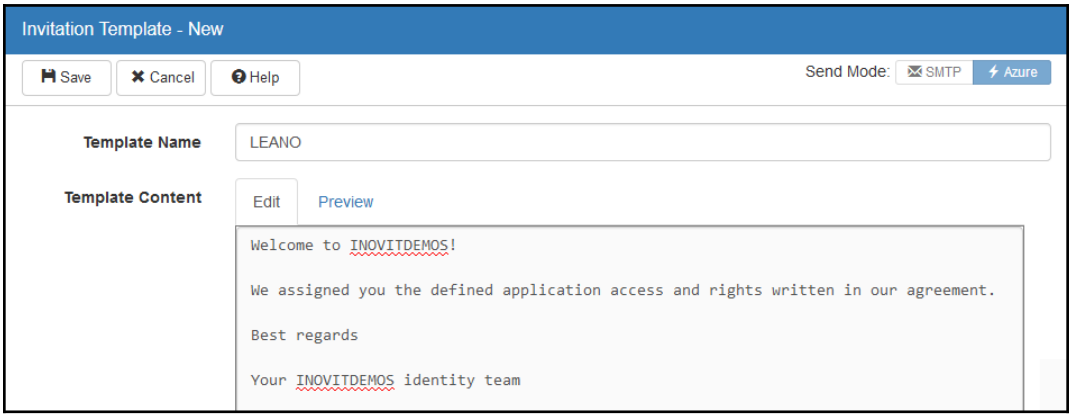
Admin options

10. A default template is already in place, as demonstrated in the following screenshot:



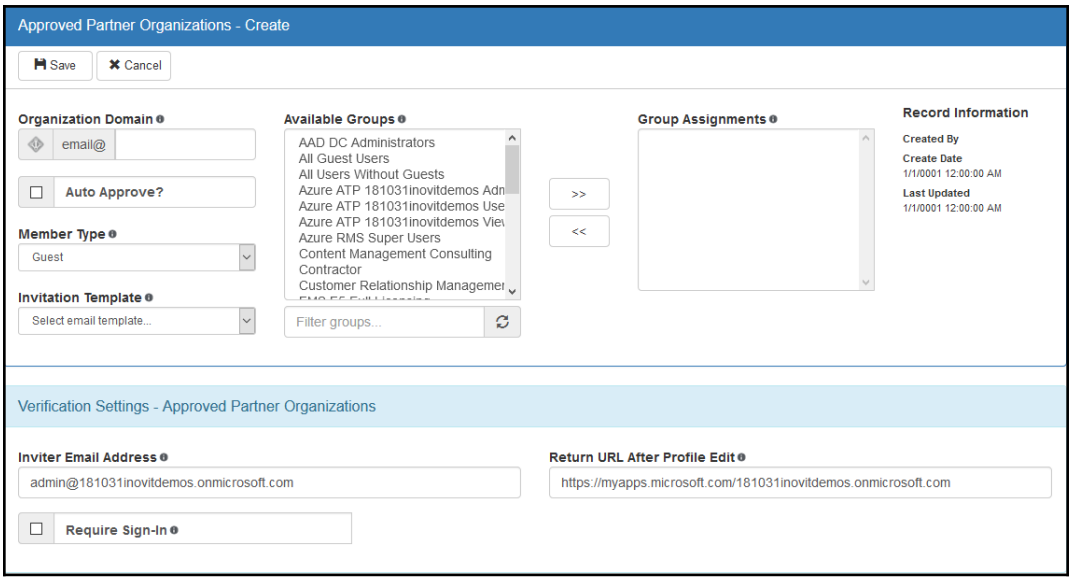
Invitation templates configuration

11. Click on **+ New** to create an additional template, as follows:



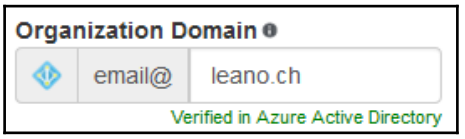
Custom mail templates configuration

12. If you want to use SMTP, your web app needs to have an SMTP server configured.
13. Click on **Save**.
14. Next, we will see domain management.
15. In this section, you can configure the main process parts, which are as follows:
    - Defining pre-approved organizations.
    - Defining the user type, **Guest** or **Member**.
    - You can use the predefined mail templates, based on the organization.
    - You can put the invited user directly into the required group, as follows:



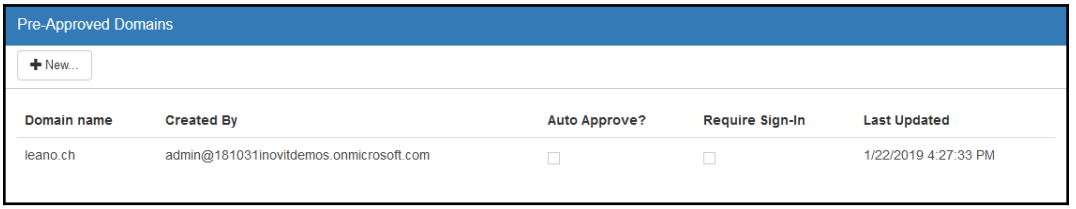
Automatic group assignment and access management options

16. Type the `yourdomain3.onmicrosoft` (`leano.ch`) organization domain.
17. Assign the **Kerberos Demo Application Access** group to the user.
18. You will notice that the application can check directly whether the organization is based on Azure AD, as follows:



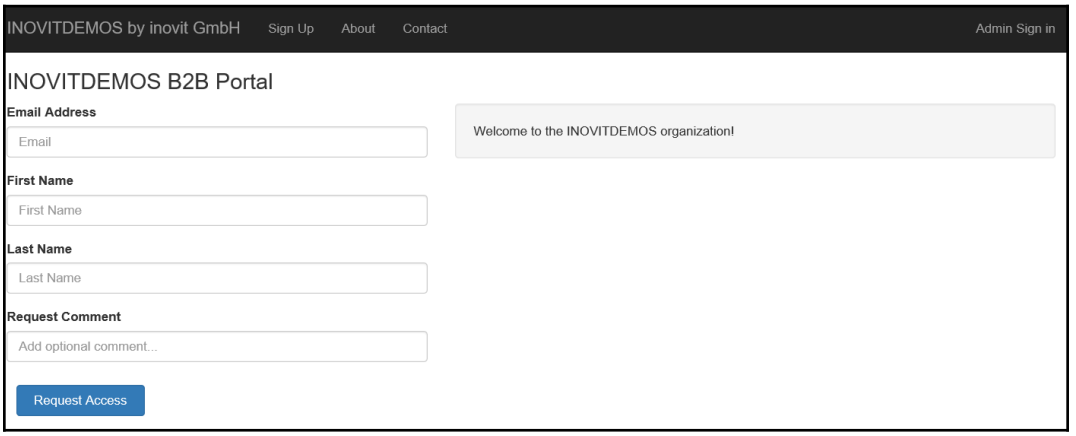
Suffix verification against Azure AD list

19. Click on **Save**:



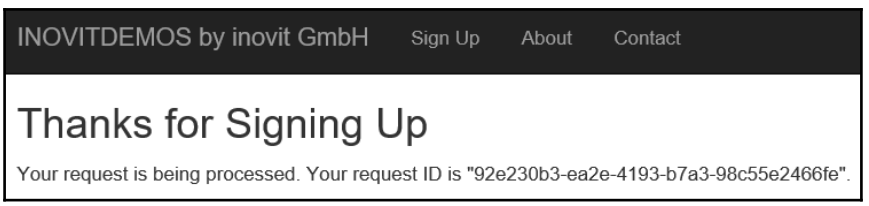
Pre-approved domain configuration

20. Now, we can test the functionality with our first guest user.
21. Open the B2B portal, as follows:



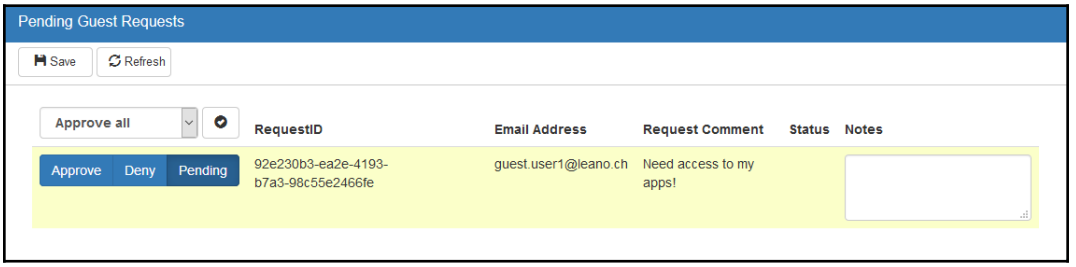
Azure AD B2B example app sign-up page

22. Sign up with `guest.user1@leano.ch`, and click on **Request Access**.
23. You will receive the following message to identify your request:



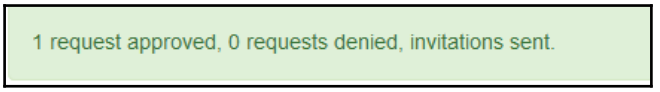
Sign up, including the request ID

24. Next, log in to the B2B portal as an administrator.
25. You will see the request from the user, as follows:



Guest approval process

26. The guest user has not yet been created in the organization.
27. Approve the request and save, as follows:



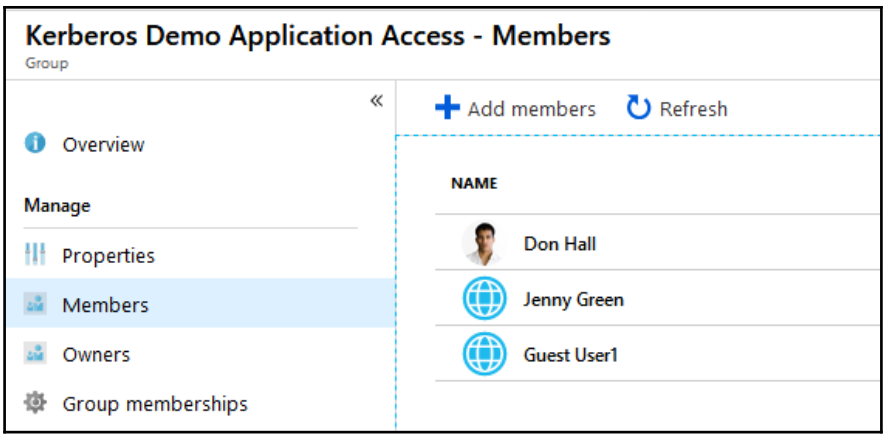
Request approval

28. Log in to `https://myapps.microsoft.com` as **Guest User1**.
29. You should get the notification about the invite, and you can perform the onboarding process.
30. If a user is invited but didn't accept the invitation, the guest user will be in the following state:

User type and state

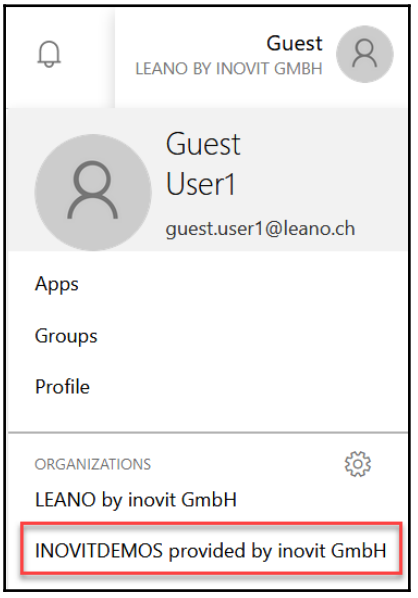
31. The state will be changed after the user accepts.
32. The following should be the expected result.

33. The user is assigned to the app group, as follows:



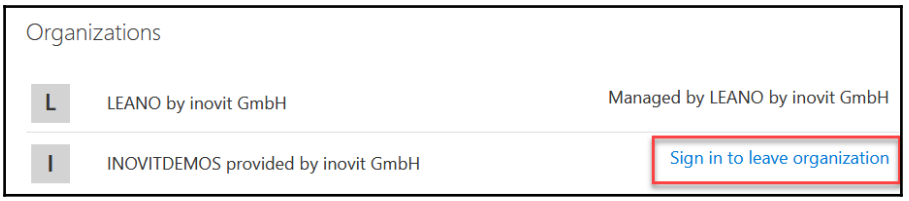
Group assignment

34. The user is a member of the organization. Click on his user icon in `https://myapps.microsoft.com`, as follows:



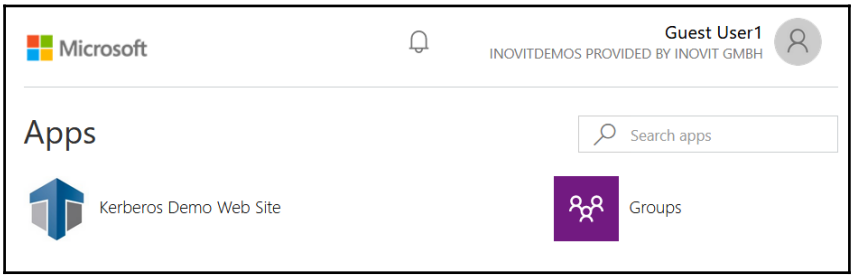
Organization association

35. If you click on the wheel, you will be able to leave the organization (feature related to the General Data Protection Regulations GDPR), as shown in the following screenshot:



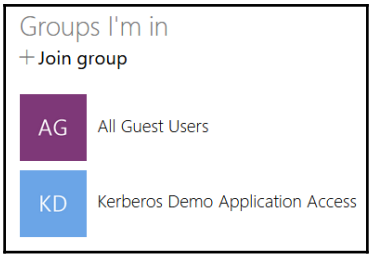
Leave organization option

36. And, if you change the organization, the **Kerberos Demo Web Site** application should appear, as follows:



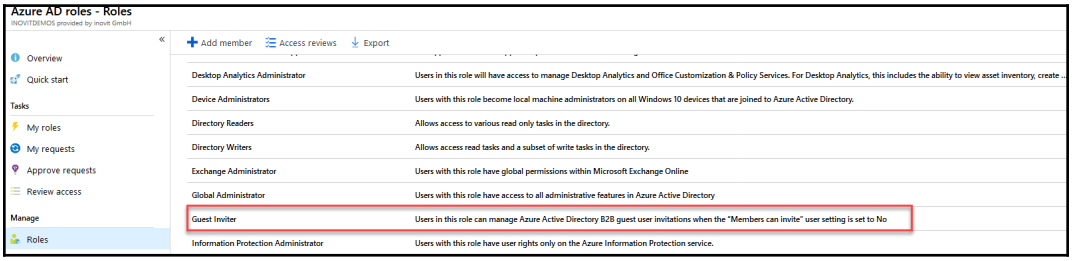
Assigned applications on the Azure AD access panel UI

37. For now, if you click on the application, you will receive an error; we will handle this later in this chapter.
38. You will be a member of the following groups, when you click on **Groups**, as follows:



Group assignment self-service and overview

39. The next process that the B2B portal supports is auto-approval for organizations that you know.
40. If you decide to delegate the invitation process to an internal or external employee over the Azure AD portal or the B2B portal, you can use the Azure AD PIM AD **Guest Inviter** role, as follows:



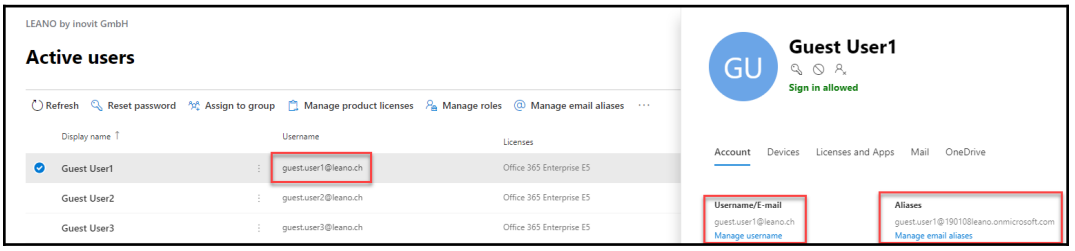
Delegation of the guest inviter role

During this use case, you have seen the power of the invitation API. You can work around it and build your own scenarios.

Special considerations

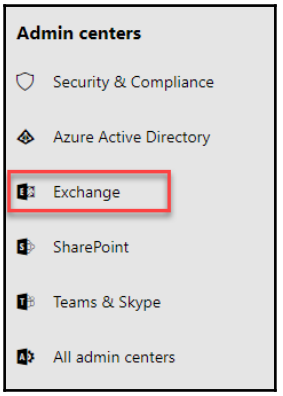
In this small section, we will take a look at some special parts of the B2B process. Let's start with the scenario that a user gets married and changes their name, including the email address and the `userPrincipalName`. Follow these steps:

1. To view the result, we will do this for `guest.user1@leano.ch`.
2. Open `https://portal.office.com` on the `yourdomain3.onmicrosoft.com` tenant and select the user, as follows:



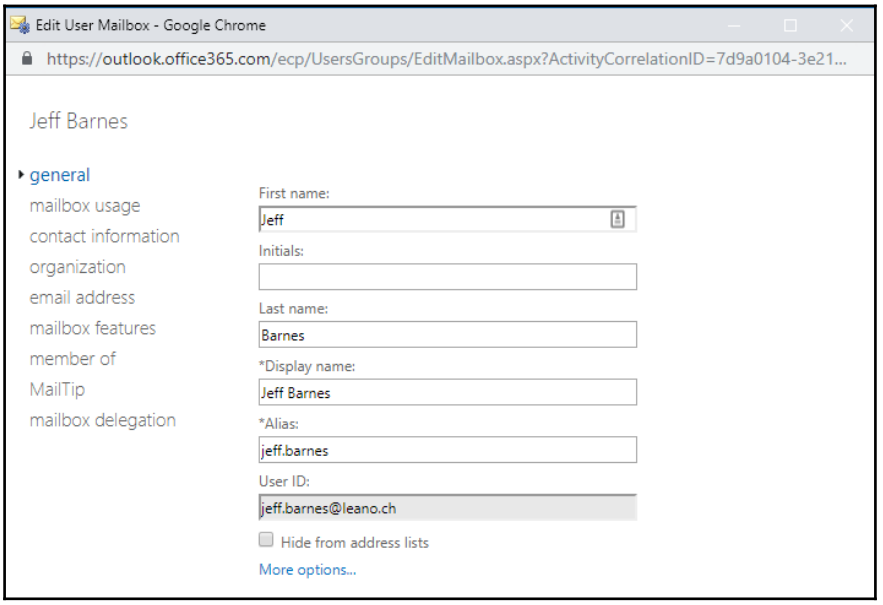
Guest user modification options and results

3. Click on **Manage user** and call him `jeff.barnes@leano.ch`.
4. Navigate to the **Exchange Admin Center**, as follows:



Exchange Admin center access

5. Navigate to **Recipients | Mailboxes.**
6. Edit the **Guest User1** new **Jeff Barnes**, with the following expected result:

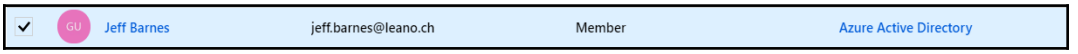


Email address change for the guest user

7. Go to the **Email address**.
8. Change all of the following to **Jeff Barnes**, as follows:

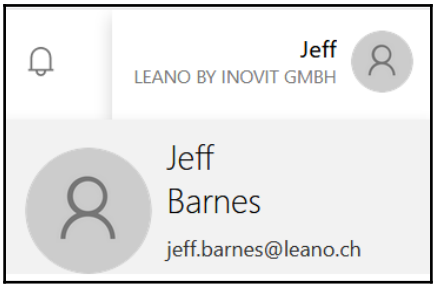
Changing the user attributes

9. Your user should now look like this in the Azure AD:



Result of changes

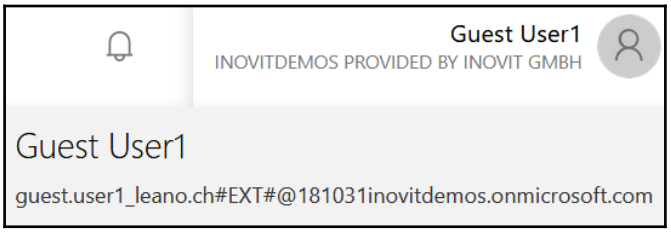
10. Next, we will test the behavior on `https://myapps.microsoft.com` with **Jeff Barnes** (**Guest User1**).
11. Your user should look like this:



Review of the changes in the Azure AD Access Panel UI

12. Now, change to the `INOVITDEMOS` organization.

13. You should have the same experience as before, because the mapping in the background is done by a GUID.
14. However, you should notice, as in the following, that the user is still called **Guest User1**:



Review of the changes in the Azure AD Access Panel UI (External org)

15. There is no synchronization job between the two organizations.
16. You can change the experience for the user by changing his information, as follows:

Attribute change options

17. There are more things that you should note about guest users, as follows:
    - They can't get Exchange Online licenses
    - They can't change the tenant in Office 365 (for PIM enabled external supporters, for example)
    - You can't leave an organization if your sign in is blocked

You can read more about the special considerations at `https://docs.microsoft.com/en-us/azure/active-directory/b2b/user-properties`, including how you can use access reviews to manage the Azure AD B2B guest users.

On-premise application access for guest users

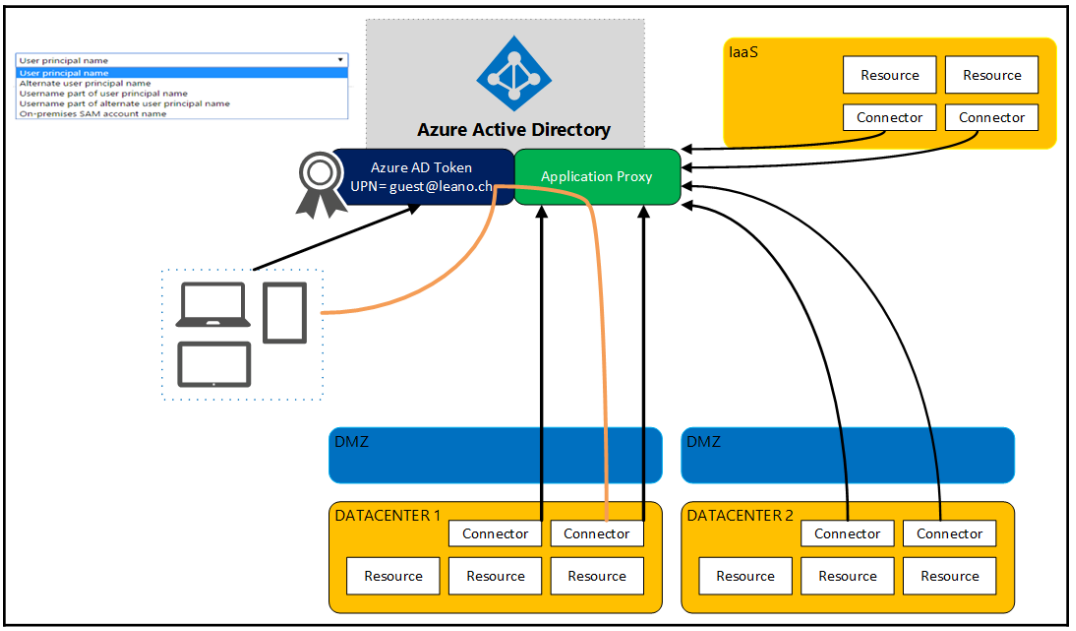
In this section, we will illustrate how you can provide on-premise application access for guest users. We have already published the Kerberos Demo Web App through the Azure AD Application Proxy, and you have seen the application in the Azure AD Access Panel UI as a guest user.

With the integration of the web application proxy and the Azure AD Application Proxy, you will receive the following features and capabilities:

1. End user portal: The Azure Access panel at `https://myapps.microsoft.com`.
2. The Azure AD authentication capabilities are as follows:
    - Usernames and passwords synced from on-premise AD DS
    - Federated login to on-premise or other federation servers
    - MFA
    - Customized login screen
    - Authorization based on the user or groups
    - SSO to Office 365, thousands of SaaS applications, and all applications integrated with Azure AD.
3. Reports, auditing, and security monitoring are based on big data and machine learning.
4. All HTTPS traffic is terminated in the cloud, blocking most HTTP-level attacks.
5. Unauthenticated traffic is filtered in the cloud, and will not arrive on-premise.
6. There are no incoming connections to the corporate network—only outgoing connections to the Azure AD application proxy service.
7. Internet-facing services are always up to date with the latest security patches and server upgrades.
8. There is login abnormality detection, reporting, and auditing by Azure AD.
9. Providing SSO experience from Azure AD to on-premise applications.
10. Connectors use the Azure AD token data to impersonate the end user to the backend applications using **Kerberos Constrained Delegation** (**KCD**).
11. There is support for any application that uses **Integrated Windows Authentication** (**IWA**), such as SharePoint, Outlook Web Access, and Microsoft Dynamics CRM.

12. There is no need to change the backend applications.
13. There is no need to install agents on backend applications.
14. There is no need to expose on-premise applications directly to the internet.

The following diagram shows the architecture:



Azure AD Application Proxy architecture

In the case of guest user access, the Azure AD application proxy just checks that the `userPrincipalName` exists in the on-premise Active Directory.

To configure the access for guest users, we will use the following procedure:

1. Log in to your Domain Controller **YD1ADS01** as a domain administrator.
2. Open a PowerShell and connect to your `Azure AD`, as follows:

```
Connect-AzureAD
```

3. Get all guest users, as follows:

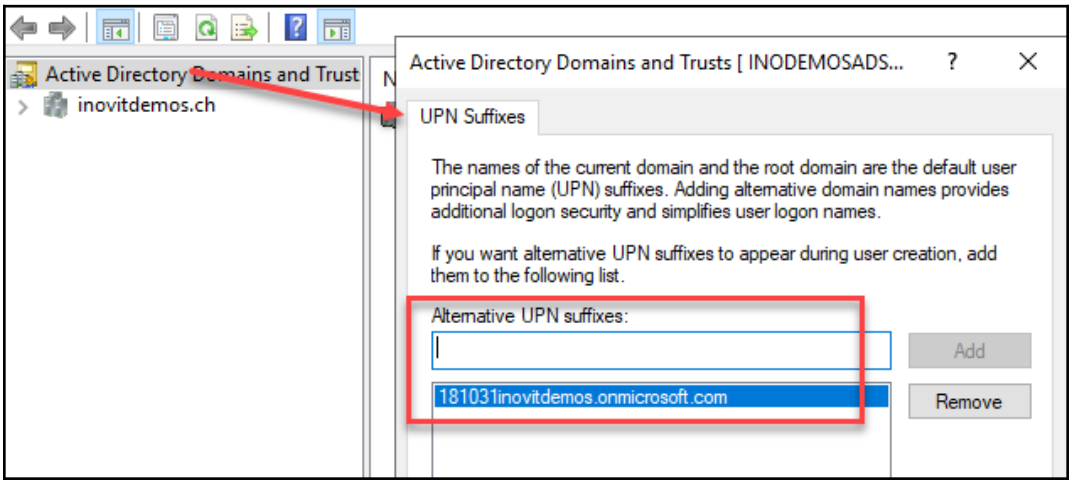
```
Get-AzureADUser -Filter "UserType eq 'Guest'"
```

4. The following screenshot shows the output:

```
PS C:\Users\cloudadmin> Get-AzureADUser -Filter "UserType eq 'Guest'"

ObjectId                             DisplayName   UserPrincipalName
--------                             -----------   -----------------
07bf426f-4188-433d-99b9-d14d2f3e62c1 Jochen Nickel jochen.nickel_inovit.ch#EXT#@181031inovitdemos.onmicrosoft.com
f0288126-8b6d-47b1-9984-acb48e66f851 jochen.nickel jochen.nickel_gmail.com#EXT#@181031inovitdemos.onmicrosoft.com
7efba417-267c-478a-b038-8161bcb0f558 Jenny Green   jenny.green_leano.ch#EXT#@181031inovitdemos.onmicrosoft.com
b3664d12-adfa-4cc1-9f90-b23ddf968e08 Susi Delgado  susi.delgado_identityplus.ch#EXT#@181031inovitdemos.onmicrosoft.com
7448d477-0aa2-4677-bb2f-3e1cff6a6c3f Maria Lee     maria.lee_inovit.ch#EXT#@181031inovitdemos.onmicrosoft.com
427314ed-5a15-4b66-b10b-312a458e7e25 Guest User1   guest.user1_leano.ch#EXT#@181031inovitdemos.onmicrosoft.com
```

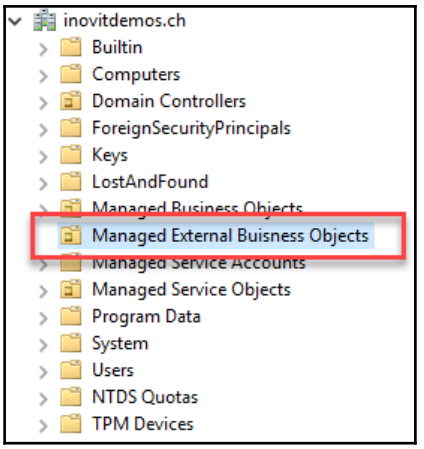
All guest users of the tenant

5. Now, we need to add the `UPN-suffix @yourdomain1.onmicrosoft.onmicrosoft.com` to the UPN suffixes of your Active Directory environment.
6. Use the `domain.msc` console for it; right-click on **Active Directory Domains and Trusts** and choose **Properties**.
7. Add your suffix, as follows:



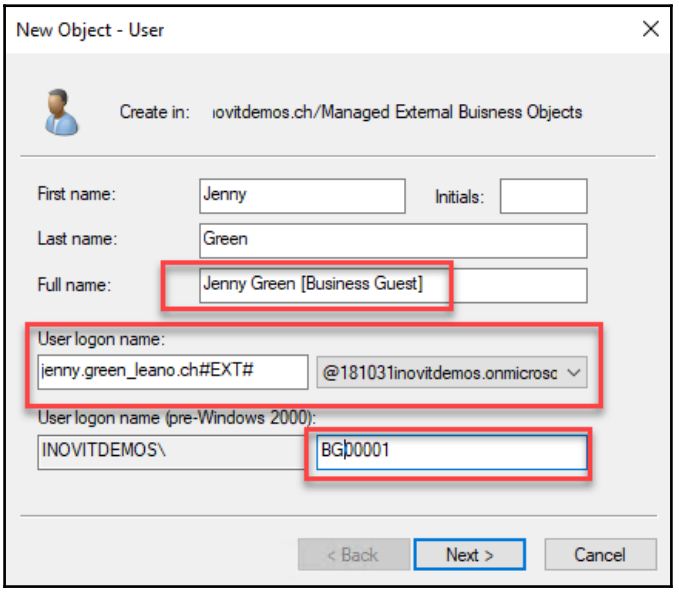
Azure AD suffix to UPN suffix assignment

8. Next, we need to create our guest user in our local Active Directory.
9. Open the console `dsa.msc` Active Directory Users and Computers.
10. Create an **Organizational Unit** called **Managed External Business Objects**, as follows:



External objects OU in Active Directory

11. Create the user, as follows:
    - `jenny.green_leano.ch#EXT#`
    - `@181031inovitdemos.onmicrosoft.com`
    - `BG00001`:

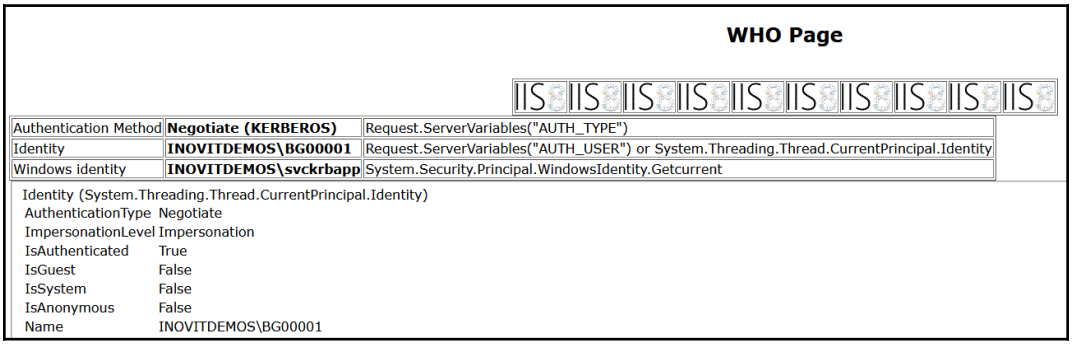


User properties

12. You don't need to remember the password, because it's not needed.
13. The Azure AD will authenticate the user.
14. Now, assign the user to the **Kerberos Demo Web App Access** group in the Azure AD, as follows:

App access group members

15. Now, you can test the solution with Jenny Green at `https://myapps.microsoft.com`.
16. Change the organization to `INOVITDEMOS`.
17. Click on the **Kerberos** app.
18. You should see a successful login, as follows:



Kerberos example web page for guest users

To automate and extend this solution, you can use Microsoft Identity Manager 2016 or the provided PowerShell solution. You can get information about this approach on Azure AD B2B collaboration with **Microsoft Identity Manager** (**MIM**) 2016 SP1, with Azure application proxy, at the following sources:

- `https://docs.microsoft.com/en-us/microsoft-identity-manager/microsoft-identity-manager-2016-graph-b2b-scenario#b2b-end-to-end-deployment-example-scenarios`
- `https://www.microsoft.com/en-us/download/details.aspx?id=51495`

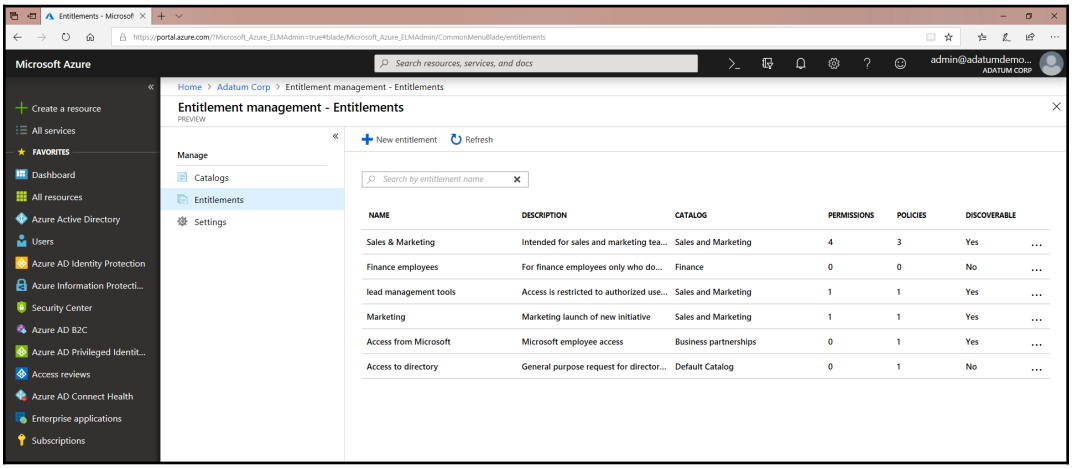
In this section, we handled various use cases to provide an identity life cycle for guest users. In the next part, we will give some tips on how you can get more comfortable in your environment.

Azure services for automation

This year, at Microsoft Ignite, Mark Wahl presented a new feature set. This feature set is called Azure AD Identity Governance, which is in private preview. I will give you an idea of the public information.

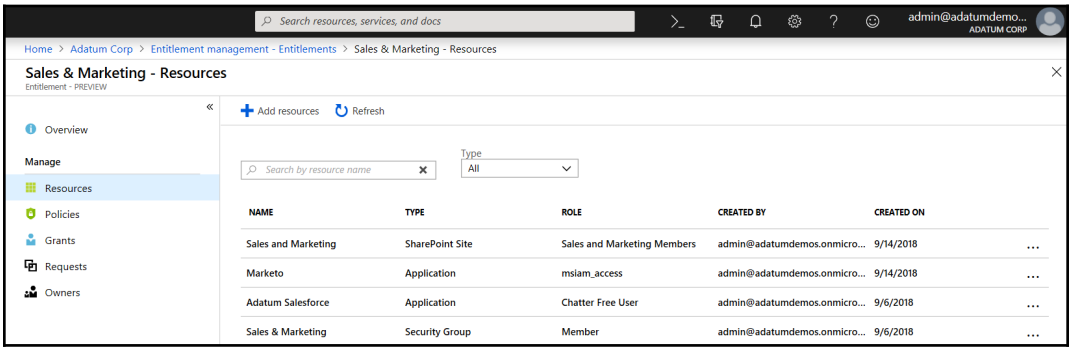
Identity is the control panel and merges user experience, business needs, and security requirements together. You need to ensure that the correct users get the right access to the correct and required resources at any time. Azure AD Identity Governance is the set of capabilities for this. They enable you to define access policies and monitor your identities. Microsoft is developing a complete suite of governance capabilities for Azure AD, including two powerful new features: **Entitlement management** and **My Access**.

Admins will be able to create policies for resources, such as groups, apps, and sites, with the upcoming entitlement management. It will provide the automated process of granting access to employees and partners. The **My Access** portal gives employees and partners the possibility to request access to these entitlements, including access to request approvals, as shown in the following screenshot:



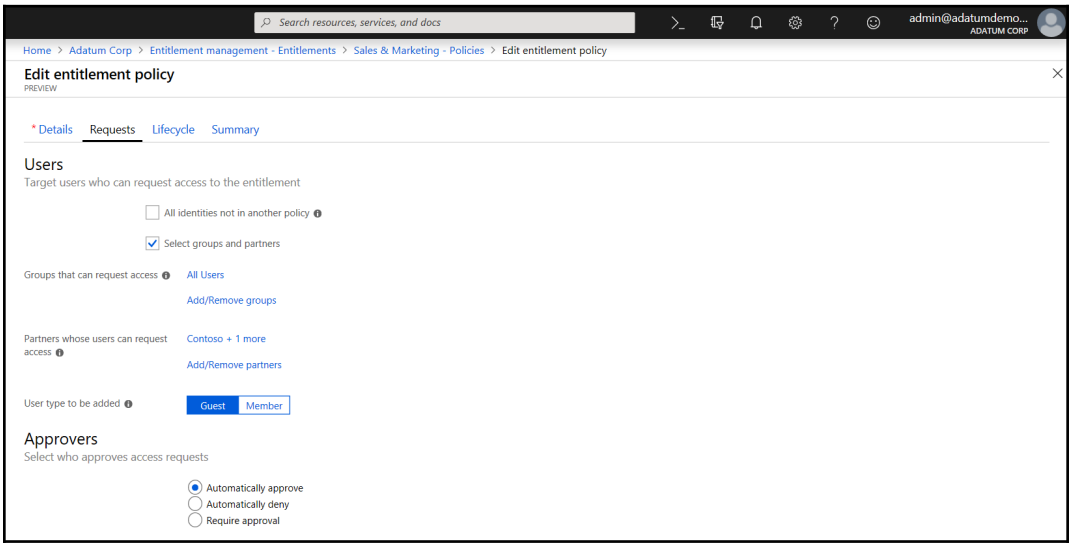
Azure Identity Governance Entitlement management options

First, the resources must be specified and associated with the entitlement. In this example, you see two apps, one user group, and one SharePoint site. You can add more resources. As you can see in the following screenshot, roles are also included in this concept:



Resource assignment for entitlements

With the new features, as shown in the following screenshot, you can define entitlement policies to drive your security requirements:



Entitlement policy options

Another fact that you should be aware of is the many different Azure features that provide you with the capabilities to drive your cloud-based identity and access management. We recommend reading the article at `https://bit.ly/2u3LcZG`, regarding the following technologies:

- Microsoft Flow
- Azure Logic Apps
- Azure Functions
- Azure App Service WebJobs

There are already many examples in the community that show the power of these features; some of them are as follows:

- Martina Grom, with provisioning Office 365 groups with Azure functions: `https://blog.atwork.at/post/2017/10/01/Provisioning-an-Office-365-group-with-an-approval-flow-and-Azure-functions-part-2`
- Asish Padhy, with the same topic: `https://asishpadhy.com/2018/05/01/automation-and-creation-of-office-365-groups-using-flow-microsoft-graph-and-azure-function-part-1/`

- Microsoft Docs, with Microsoft Graph bindings for Azure Functions: `https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-microsoft-graph`
- Microsoft, with an introduction to Azure Functions: `https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview`

We highly recommend trying to work with these technologies, because they will be the future of providing implementation to our customers.

Summary

Working through this chapter, you saw how you can build a rich Azure B2B solution with an identity life cycle, including on-premise application access for guests. Furthermore, you also had an excellent introduction to the possible use cases and gaps. We also provided an overview of the upcoming Identity Governance features that Microsoft is developing, and the power that they will bring to the game; for example, there is a resource assignment based on roles and associated policies.

In the next chapter, we will deploy single-tenant and multi-tenant applications to your environment, and will provide an introduction to these concepts.

3
Section 3: Data Classification and Information Protection

In this final section of the book, you will learn to understand the key principals of a security culture and data classification. Furthermore, you will learn to plan, implement, troubleshoot, and develop solutions on current Microsoft Information Protection technologies like Azure Information Protection, Cloud App Security, and Windows Defender ATP.

The following chapters will be covered in this section:

- `Chapter 12`, Creating a Security Culture
- `Chapter 13`, Identifying and Detecting Sensitive Data
- `Chapter 14`, Understanding Encryption Key Management Strategies
- `Chapter 15`, Configuring Azure Information Protection Solutions
- `Chapter 16`, Azure Information Protection Development

12
Creating a Security Culture

Organizations need to build a security culture to provide a suitable information-protection solution. In this chapter, you will get an overview of the four main pillars of security culture, which are leadership support, efficient training, ongoing testing, and continuous communication to the entire organization and its partners. If you don't establish a security culture, you will have difficulty being successful in every part of an information-protection strategy, because every employee needs to know what information needs to be protected. Furthermore, the introduction of security measures can result in high costs if they are not sufficiently planned and not supported by the management.

An additional focus in this chapter is data classification, as the classification of information provides the basis for most security mechanisms. The classification of information provides context for the factors that lead to security policies taking effect and triggering the right level of protection. Also, we will explain how applying the four main parts of a security culture leads to supporting the introduction of data classification. Finally, we will have our first contact with the Microsoft classification and protection solution to introduce the practical application in the following chapters.

This chapter covers the following topics:

- Why do we need a security culture?
- The four main pillars of a good security culture
- General overview of data classification
- **Azure Information Protection** (**AIP**) overview

Why do we need a security culture?

The ideas, duties, and behaviors of a group of people influence their security and can put them at risk, hence there is a need for security culture. In the professional world, security culture is used to describe the kinds of behaviors organizations would like to see in their employees in areas such as cyber, physical, and personal security. Security training and the change of the behavior of a co-worker to adapt to the safety guidelines of the enterprise are necessary. The main focus of a security culture is keeping infiltrators or other potentially damaging parties out. In our case, we will focus on the relevance to data security, because to handle the complete field of security goes beyond the scope and focus of this book. So, let's see what this means for us.

Digital transformation has reshaped the way we work. Data has become the new currency of business. Information such as development results, intended company acquisitions, sales account information, personal or financial data, and security information have very high values in organizations. A lot of the information and data created or exported from systems flows into and through organizations, out into the cloud, and across the Internet of Things. More and more data is in motion, and this creates more and more risk. This fact leads organizations to make extensive investments in data security, to protect their intellectual property and the trust of their customers.

Technology alone doesn't solve the problems in the field of data security. Furthermore, if you think data security is only an IT task, you will remain vulnerable to breaches caused by user activity. Many security professionals still focus too much on processes and technology, or just on technology. This fact puts organizations in an unbalanced and weak position. We need to include all the employees in our strategy because if we don't, we leave the organization vulnerable to an attack factor that's nearly half of the current breaches in the actual cyber security world. In an enterprise, the elements, people, processes, and technology must be in harmony to create a strong foundation for a proper security culture, a successful data security strategy, and digital transformation.

In organizations with a proper security culture, employees have a clear understanding about what is right and wrong, the type of activity they should report, and how to contact the right team. On the other hand, in organizations with a weak security culture, security becomes someone else's responsibility. A vital fact of an effective security culture is to have a shared sense of ownership.

So, it's quite important to understand that you need to engage and educate employees to make better decisions on how information is handled. By prompting your users to pause and consider the business value of the information they own and control, you're empowering them to use it in a way that won't put your corporate data at risk. The use of data classification enables employees to handle sensitive content correctly.

Already during the creation of information, or when saving, the employee is supposed to categorize the information correctly and to trigger the right protection. Besides, the employee learns to better assess the need for security and the value of the information for the company in the future, as well as to adapt their handling of sensitive content to the organizational guidelines. The data classification additionally provides the factors that are essential for security reports. It will also trigger the execution of security functions at the right time to keep the business in a secure state.

You can invest a lot of money and energy in improving your security processes and technology controls. If you don't include your end users within your organization, the risk of data breaches will still be high. The most important task you need to work on is educating and empowering your users to make qualified decisions in handling the information of their daily work. To provide an appropriate security solution, you need to give a natural flow for the end user in their daily work and apply and enforce policies to avoid the violation of security rules.

By helping users to classify and give value to the information they create, they become active participants in the protection and security of data.

In the next section, we will discuss the four pillars of a security culture.

Pillars of a good security culture

Now that we have highlighted the need for a security culture, let's look at its four main components. As we have already mentioned in the introduction, we will discuss the following four areas, which are the cornerstone of a healthy and sustainable security culture:

- Leadership support
- Training
- Testing
- Continuous communication

You will get an overview and tips that will help to you install a high-quality security culture in your organization. Keep in mind that an organization's security culture is the foundation of any security control and should be supported by an information-protection strategy. In the next part of this section, we will start with the leadership support you need to support an information-protection strategy.

Leadership support

An information-protection strategy and the associated security culture always has to start with the top-level executives and board members. This will provide the funding for appropriate resources to develop, implement, and maintain the information-protection strategy, which includes the security culture.

Making security a part of everyone's responsibilities is key to a living security culture and reliable data security. For example, if a data-classification project is launched without management support, there is a high probability that the project will fail. Often, the reasons are that the necessary resources are not available or that processes and business alignment have changed.

Your executive sponsors should continually be kept up to date on the progress, because they will be ready to address any significant blocks. Solutions and initiatives should always be aligned with the business goals and the processes of the organization. Furthermore, you should engage and establish department/team champions so that they understand and support your strategy and implement it at a very early stage. They will also be an excellent resource for feedback to improve your solution.

We often see that the efforts to develop and implement an information-protection strategy and security culture falls into the hands of the IT department. As we already know, we are not talking about a purely technical solution. Complete success relies heavily on the users of an organization and their activities. Finally, don't forget that the executives should lead by example, because the rest of the organization will look up to them. This means that the executives should be an active part of pilots and live by the new strategy and culture. Be aware that they need to have special support during the implementation.

Training

As we mentioned in the *Why do we need a security culture?* section, it's essential to engage and educate employees to make better decisions in the handling of information. An appropriate data classification software can help you to address the engagement and education of your employees. Here are various effective training strategies to address the critical ones:

- Educate users and communicate what you are trying to classify and protect.
- Provide information that helps the user to get a better understanding of what is right and wrong.
- Use a mentoring approach rather than a punitive one.
- Use easy and natural language.
- Ensure your feedback loop is in place.
- Decide the training needs as per the role of the user and the associated risk, such as data or image loss of the company if the employee leaves or incorrectly handles information.
- Your information gathering doesn't need to be perfect; try to get as close to 100% as possible.
- Use an iterative approach while making improvements.

Further, you need to align the training needs to the associated risk, such as data loss of a specific user group. For example, data classification is a typical general topic everyone should be trained in. Let's think about an engineer; they should have an intermediate level of training in how they need to handle their results. The administrator who manages all the results of the engineers of an organization should have an in-depth training level, because they are maintaining all the jewels of the organization. So, they need to know exactly what to do to secure the information.

In essence, data classification will help your organization to improve its culture of information-security awareness significantly. So, invest in supporting and training your users in that field. Take feedback from users, listen to them, and continually improve your training material.

The following training approaches could be used:

- Reference posters or cheat sheets a user can keep on their desk
- Training videos that are placed on internal portals often used by your users
- A corporate video with an executive about the importance of the information-protection strategy and the security culture, which shows the leading-by-example approach
- Online forums or knowledge systems that are used inside the organization

Keep in mind that users may consume and use the training materials in a different way, depending on their role within the organization. In the next section, we will describe the testing needs inside a security culture and the implementation of an information-protection solution.

Testing

Active testing and monitoring of the information-protection strategy is crucial to keep the security culture alive. In the case of data classification, the primary goal is to achieve user acceptance of your defined data-classification labels and policies to ensure that users do not resort to cheating. To make sure that you are headed in the right direction, you need to define specific test scenarios aligned with your critical success factors in order to validate and satisfy the significant business and functional requirements for your classifications and policies.

It should also be noted that often only a small part of the information you communicate to your recipients is clearly understood. Due to this, it is essential to check global communication in small test groups that can be adjusted if necessary. Furthermore, build active monitoring of the information-protection activities and reference it to the context of the information, so that you can test the accurate classification and applied protection of the data. Your technical solution should provide you with a useful toolkit to achieve this target.

Continuous communication

An essential tool for transforming user behavior is continuous communication, starting from when you begin making changes to the organization's culture. In the beginning, it is necessary to communicate your mission to all your users. By communicating clearly, everything is aligned with the core values and the corporate commitments of the organization, and how the information protection strategy and security culture stick to these commitments – users can see the bigger picture of the security investment and can identify themselves as an essential part of the whole solution.

Let's also discuss the "what's in it for me?" factor. For example, employees need to make sure their work is aligned with the information security policies they signed on day one of their employment. For an active and thriving communication process, it's essential that you are approachable and share the improvements that you get from user feedback.

General overview of data classification

After we describe the necessity and the four central elements of the security culture, we will discuss the data classification to create the basis for a successful solution in the area of information protection. As already mentioned, data classification provides the necessary information and tools for the correct use of protective measures. Let's start with a basic definition of data classification. It is a continuous process of consistently categorizing information based on specific and predefined criteria so that the data can be efficiently verified, effectively identified, and protected. This is why data classification is the foundation of data security.

Successful data classification requires a broad awareness of the needs of an organization and a thorough understanding of where the data assets are located and how data or information is generated. For this reason, finding and identifying existing data on various endpoints, such as network shares, databases, or cloud services, is closely tied to classification. Even when creating new data, classification is always carried out.

In addition to the different endpoints at which data is usually available, there are three basic states in which data can be located: in idle mode, during processing, and during transport. All three states require technical solutions for data classification. However, the principles used are the same for all three states. Unfortunately, data classification must always be done in the creation process. For example, confidential data about all the three states must be treated confidentially. Furthermore, a distinction has to be made between structured and unstructured data. In typical classification processes, structured data in databases or tables is less complicated than unstructured data such as documents, source code, or email messages.

Overall, it is crucial to manage unstructured and structured data based on their sensitivity level. A correct implementation of a data-classification solution leads to the creation of sensitive and confidential data with proper foresight. It can be decided whether information can be made publicly available or not.

In discussions with our customers, we have often heard about their reluctance to adopt data-classification solutions. However, such comments can often be refuted quickly. The following three examples exhibit the crucial aspects of data classification:

- **Data classification is too complicated**: In fact, many data-classification projects fail or get stuck because of classification schemes that are too complex. The data-classification schema provides different labels to apply the classification to the information. For example, the Confidential label is used for sensitive information that should not be accessible by unauthorized users. Often, we find concepts that use labels the users are not able to understand, or labels that conflict with other labels. Consequently, it is advisable to start an iterative process and only with the necessary options.
- **Data classification is just another step I need to take, and it makes me slow**: Classifying data can simplify data protection. Knowing what information is confidential allows you to choose a suitable location to store the data or adopt a protective measure. It's important to understand why and how data is protected. Furthermore, data classification has the advantage that data can be found more easily and the employee can also protect themself by classifying the information correctly and applying the appropriate protection.
- **Data classification pays off only after a long time**: Automated classification can provide insights from day one. The automation of the two areas of context and content can be quickly and easily created in a certain order and additional information can be obtained to promote safety improvements. You will find out more about the different classification methods in the *Methods of data classification* section.

Now that we have established a basic definition of data classification, let's discuss the different methods of data classification.

Methods of data classification

In this section, we will work through different methods of data classification, explaining each process and providing the essential arguments for the methods, including what information will be gained for the classified information. To understand these methods, it is essential that you are able to define the correct and most efficient way to your classification system. There are two basic methods of classifying data: manually through interaction with the user, and automatically through classification software based on a set of rules.

For data classification, the following four methods are usually used:

- **Based on the user's choice (manual)**:
    - Focus on the manual selection of the user
    - Should be combined with an automatic process based on content and context that recommends the appropriate classification
    - Reclassification should be controlled and monitored
    - Based on the knowledge of the user when creating, editing, or reviewing sensitive information

  Information gained: What is the content and context of a document?

- **Based on the content (manual/automatically)**:
    - Focus on examining and interpreting sensitive content using regular expressions, metadata, or other options

  Information gained: What is the content of a document?

- **Based on the context (manual/automatically)**:
    - Focus on the application, location, or creator, as well as other variables and indicators of sensitive information

  Information gained: Who accesses it? When will the data be accessed? Where is the data moving? How is the data used?

- **Supported by machine learning (manual/automatically)**:
    - Focus on document classes around automatic classification by comparison with comparison sets and training

Information gained: System learns to integrate further options into the classification or to increase the recognition of certain information.

The combination of different classification methods combines the different positive properties of the individual methods, as follows:

- Metadata-based classification allows exceptionally high speeds
- Pattern-matching is fast and safe for classification criteria that can be identified with regular expressions
- Machine learning with linguistic-statistical methods is generally applicable and gives very good results with fuzzy criteria
- Special classifiers, such as image recognition, can be another method

The combined and continuous use of these methods leads to success.

Now that we know the different classification methods, we can address the challenges facing organizations where the use of data classification can support a fundamental improvement in the management of information and the efficient use of security technology.

Data classification and unstructured data

An essential aspect of why data classification is necessary is the amount of information generated within a company. The amount of information is often unmanageable and unusable. The management of unstructured data is still one of the larger unresolved issues of storage technologies. Still, about 80% of the stored data is probably unstructured, which is beyond the capabilities of an economical and sustainable information-management process. However, the critical challenge of process-aware allocation exists in both structured and unstructured data.

Furthermore, the current data volume does not indicate any externally visible information about the content and the intended use. If we use a classification scan with a few simple keywords or analyze the metadata, there can be more transparency of the different types of data. The results are gaps in the management of the company's information, which should be identified by the following characteristics:

- Sensitive information that needs special protection
- Important and/or regulated information that needs to be archived
- Passive information that is seldom needed
- Superfluous information that could be deleted
- Technical log information with special treatment

For this reason, the recommended way of creating a classification is to create the necessary transparency about the contents of the company's-relevant information.

In addition, the following advantages also result from the data classification:

- **Classification**: Allows a company to group data consistently across its business functions, regardless of its format or delivery system.
- **Storage Optimization**: Gives an introduction of a storage classification concept (tiered storage model).
- **Data value**: This allows data to be differentiated according to its value for the company.
- **Archiving**: Introduction of an archive for better information retrieval.
- **Transparency**: Information type and relevant attributes are visible from the outside.
- **Synergies**: Users and applications can use the information.
- **Safety**: Efficient and sustainable protection mechanisms can be controlled.
- **Efficiency**: Proof of information in processes and groups for processing.
- **Unity**: In the company or organization by following the same principles.

Now that we have seen the advantages of data classification in the field of unstructured data, we can view the use case of data leakage and loss prevention.

Data classification and Data Leakage/Loss Prevention

Data Leakage/Loss Prevention (**DLP**) is the term used to denote solutions designed to prevent the uncontrolled outflow of information from the company. DLP is one of the IT measures for risk management. Today's DLP solutions provide only a fraction of their potential, because the rule definition for the permission and prevention of access by filing structures and other technical attributes is complicated. Since today's solutions often lack the appropriate methods for integrating the specialist pages, the specialist departments often do not feel responsible. Thus, DLP remains in many cases a blunt sword in the fight against the unauthorized outflow of information.

With the introduction of the manual and automated classification of all files, DLP can act on the classification rather than having to be painstakingly managed through the filing structure. You will find the information about the different classification methods in the *Methods of data classification* section. This is visible in frequently changing structures in the filesystem. The rule definition is intrinsically useful across the enterprise, and the administration expenditure sinks drastically. The concept of classification and document classes makes it much easier to integrate the specialist pages.

In risk management, too, the efficiency of already established solutions, for example in the area of DLP, can be increased many times over by using a classification.

Next, we will focus on data classification and compliance aspects.

Data classification and compliance

Archiving is key to meeting many compliance requirements. Today, many important pieces of information that need to be archived from a compliance point of view remain unarchived because they are not recognized as requiring archiving by the system or process. An example of this is the many contract-relevant files that end up in the filesystem. In particular, the large stocks of data on file servers represent a black box for many companies, in which the relevance for specific compliance issues is only partially known. Through the automated classification of all files in the company, relevant information is recognized and archived by the classification. Through the cross-company approach and automation, a significantly higher degree of completeness is achieved here.

Next, we will discuss how data classification can help to optimize your storage usage.

Storage optimization

Hierarchical storage management or tiered storage is the optimization of storage costs by using different performance levels to reduce the use of expensive storage systems. Files are moved from the cost-intensive systems to the lower-cost system according to different selection criteria, without users or applications noticing.

Traditional solutions typically work exclusively with technical attributes, such as file type or age, to decide on the ability of a file to migrate to less expensive storage. They usually provide too rough a grid of selection criteria to be really efficient in practice. Furthermore, there is a lack of a suitable approach to get information about the swap capabilities of file stocks from the trade pages.

The professional classification of the file and the assignment of document classes, such as an invoice, employment contract, or construction plan, provide a better way to formulate outsourcing rules. Furthermore, these rules are entirely independent of the actual structures on the file servers and therefore much easier in administration.

Classification allows storage management to be much more efficient by managing files based on their value to the business.

Previous mechanisms for access protection were based on filing structures such as folders and shares. Thus, different views cannot be displayed. There is a conflict between high administrative effort on the one hand and uncontested need for protection on the other. Also, this method is entirely dependent on the correct behavior of users; if they misplace information, the information is visible to unauthorized people. Ultimately, files are therefore insufficiently protected today.

Now that we have worked through several use cases where data classification can be helpful, we will discuss the related information about the relationship between data classification and the access-control mechanisms.

Access control to data

In terms of data classification and subsequent information protection, the authentication of a user is upstream. Authentication of a user typically consists of two parts—the username or ID to identify the user, and a password to confirm that the user's credentials are valid. Authentication does not provide the user with access to information or services. It is just a verification that the user is who they claim to be.

After the authentication of the user takes place, authorization is the process of giving an authenticated user the ability to access an application or information. Assigning the rights assigned to an authenticated user to access the information requires attention to data classification. Due to the upstream authentication, it is necessary to gain a better insight into which sensitive data is affected in the case of compromise. The following list provides some of the most critical data examples that can be affected:

- User credentials and passwords
- Credentials and passwords of privileged users
- Identifiable personal information of customers
- Financial data from customers
- Intellectual property
- Personnel/employee data
- Financial data of the company
- Medical information
- **Internet of Things** (**IoT**) or sensor-related data and metadata
- **Industrial Control System** (**ICS**) command and control data

The most significant impacts of attacking an organization are in the area of corporate trust, legal concerns, and brand reputation. The following list provides the critical areas of an organization that can be affected:

- Enterprise customer confidence
- Brand reputation
- Direct loss of sales
- Legal
- Improvement of protective controls (costs)
- Financial loss
- Regulatory fines (compliance)

Data classification may be required by a variety of factors, such as specific regulation, **General Data Protection Regulation** (**GDPR**), Federal Act on Data Protection DSG, **Health Insurance Portability and Accountability Act** (**HIPAA**), and **Payment Card Industry Data Security Standard** (**PCI DSS**), or intellectual property protection.

Further examples and their associated areas are as follows:

Area	Regulation
Conformity	PCI DSS, HIPAA/ **Health Information Technology for Economic and Clinical Health** (**HITECH**), **Gramm-Leach-Bliley Act** (**GLBA**), **International Organization for Standardization** (**ISO)** 27001, **Personal Information Protection and Electronic Documents Act** (**PIPEDA**), GDPR
Data protection	**Personally Identifiable Information** (**PII**), **Protected Health Information** (**PHI**)
Intellectual property	Copyright, trade secret, patent application document, patent accompanying document, not IP
Export compliance	**International Traffic in Arms Regulations** (**ITAR**), **Export Administration Regulations** (**EAR**)
Department	Law, human resources, finance, engineering, sales
Project	Code names for new products and other projects
Law/traceability	Privileged, hold
Maintaining	Retention time

Now that we have discussed the data-classification-related information of access controls and regularities, we will take a look at an example classification scheme and policy.

Classification scheme and policy example

The following section describes an exemplary classification scheme and associated guidelines to clarify the activities and processes involved in data classification. The following classification scheme represents a grading in terms of sensitivity and provides information on the expected effects and any necessary encryption of the information.

To be successful with your organization in an information-protection deployment, the basic requirement is to pick well-standardized labels and a common language to connect with the data-security-specific terms.

The following table shows the example classification scheme, which starts on the left with the grade of sensitivity to the data, followed by the classification label and the example description with the impact to the company and a usage scenario for how the label is used:

	Classification	**Description and usage**
Low sensitivity	Unclassified	Information that has no organizational value or sensitivity.
	Personal	Non-business data, for personal use only. Not monitored by the company. This label is especially needed if the user is allowed to create/send personal and business information.
Low sensitivity	Public	Information that does not endanger the organization when exposed outside its organizational boundaries. Approved for public usage: • **Impact**: Irrelevant • **Critical**: Irrelevant • **Encryption**: None
Low sensitivity	General	Business data that is not intended for public consumption. However, this can be shared with external partners, as required: • **Impact**: Low, up to $25,000 • **Critical**: Low • **Encryption**: If required
Medium sensitivity	Internal	Sensitive information that can be freely distributed within the organization but cannot leave the organizational boundary: • **Impact**: Low, up to $100,000 • **Critical**: Low • **Encryption**: If required The label should be used in combination with a **General** label or replaced by it, because modern organizations often work with outsourced departments.
Medium sensitivity	Confidential	Sensitive information that must remain within organizational limits and should only be shared with a limited group of internal people when needed: • **Impact**: High and important, up to $1,000,000 • **Critical**: Medium and High • **Encryption**: Highly recommended

High sensitivity	Secret	Highly sensitive information that, if displayed outside of the organization or inadvertently to internal staff, can result in significant organizational risk: • **Impact**: Very high, > $10,000,000 • **Critical**: Very high • **Encryption**: Yes High Confidential or Restricted is also used instead of Secret. Sometimes Restricted is not completely clear for everyone, and confuses the order of the labels: Confidential and then Restricted or vice versa.

Now that we have an overview of the classification scheme, let's jump into more details with examples of the affected information by a classification label.

Description of the classification scheme

The following classification labels and associated guidelines explain the example classification schema from the previous section, *Classification scheme and policy example*:

- **Public—General information**: Company information expressly approved for publication.
    - **Examples**: Marketing material, published quarterly results.
    - The authorization according to the corporate communication policy should take place before the classification. Until approval is granted, information will be classified as Secret, Confidential, or Internal. There are no other information security restrictions for public information.
- **Internal—Internal information**: Information that can be shared across the enterprise but not for release outside the company. Often the standard classification for company information.
    - **Examples**: Department announcements, project reports, employee contact details.
    - Internal information may only be shared with employees, contractors, or business partners. Access should be from proprietary IT systems using enterprise-approved applications and services.

- **General—General business information**: Business data that is not intended for public consumption.
    - **Examples**: Company's internal telephone directory, organizational charts, internal standards, and most internal communication.
    - However, this information can be shared with external partners, as required.
- **Confidential—Confidential information**: Information that, if compromised, may harm the company's interests or cause significant disruption or embarrassment to the company or its employees. Access is therefore restricted to authorized persons or groups only.
    - **Examples**: Personal data, contract negotiations, trading strategies, upstream technical data.
    - The following restrictions apply to secret information:
        - It must clearly indicate the classification in the header or footer of each page or slide.
        - A need-to-know practice must be known and implemented in the case of contractors and business partners where a confidentiality agreement exists.
        - Store the information on corporate IT systems and access the information only through enterprise-approved applications and services.
        - Storage on portable devices or removable media is only with an active encryption.
        - Encrypt the information when sending email or external exchange, as encryption of email messages is a good safety practice.
        - Confidential information may only be shared or stored in the business-collaboration application, where access is restricted to an authorized person or group.

Keep in mind that everyone should have a common set of labels. Don't make it too complicated. If you have special needs, such as top-secret projects, acquisitions, or mergers, scope the labels to the correct group of users.

- **SECRET—Secret information**: Information of high value or sensitivity that could cause a serious corporate group compromise. Access is therefore limited to a small number of high-trust individuals.
    - **Examples**: Acquisition plans, unpublished company results, preliminary assessments of economically sensitive exploration opportunities.
    - The following restrictions apply to secret information:
        - Display the classification in the header or footer of each page or slide, or in the first line of an email (but not in the subject line). Display the name and title of the information owner and a SECRET disclaimer on the first page or slide.
        - Sharing information only where "need-to-know" exists, and most of all only with trusted people from the "in-the-know" list managed by the information owner.
        - Store information only on corporate IT systems and access by using applications that are allowed for secret information. For example, these applications have multi-factor authentication and encryption.
        - Secret information should not be included in emails to external parties unless they are appropriately encrypted. Instead, an email with a link to a saved file should be sent to an authorized system.
        - Secret information should only be shared internally via links to files stored on an authorized IT system. If not possible, an encrypted email can be used internally.

Now that we've discussed the different classification labels and their usage, we can work through some visual marking examples based on the example classification schema.

Visual markings and rules based on the classification label

The following table shows the guidelines that are applied to the information, based on the classification scheme and safety information. A subdivision into documents and messages is made.

Here, you can see the different visual markings and rules in documents and messages based on the classification schema:

Affected labels	**Visual marking**	**Visual marking and rule details**
All	Classification metadata is applied.	Classification metadata is applied.Messages cannot have a classified attachment with a higher sensitivity than the message.
Unclassified	No visual marks.	No visual marks.Messages or attachments must not contain sensitive text.
Personal	No visual marks.	No visual marks.Messages or attachments must not contain sensitive text.
Public	No visual marks.	No visual marks.Messages or attachments must not contain sensitive text.
General	Header/footer with the word **General** in Arial 12, bold, centered, light gray.	Header with text **Classification: General**, in blue.Messages contain sensitive information or text.
Internal	Header/footer with the word **Internal** in Arial 12, bold, centered, light gray.	Header with text **Classification: Internal**, in green.Recipients may only be in the `@inovit.ch` domain.Messages or attachments must not contain sensitive text.
Confidential	Header/footer with the word **Confidential**, in Arial 12, bold, centered, light gray.	Header with the text **Classification: Confidential**, in orange.The message must be encrypted with Microsoft Azure RMS.Footer with the following text: This information is considered to be inovit Confidential. Proper handling and limited distribution must be enforced. For more information, see the policy at the corporate website.

Secret	Header/footer with the word **Secret**, in Arial 12, bold, centered, red. Classification downgrade is not allowed.	Header with the text **Classification: Secret**, in bold and red.Footer with the following text: This information is considered to be inovit Secret. Proper handling and limited distribution must be enforced. For more information, see the policy at the corporate website.Recipients may only be in the Active Directory group inovit Executives.The message must be encrypted with Microsoft Azure RMS.Downgrading the classification is not allowed.

Now that we have seen the visual marking and rule examples, let's follow up with an example of general desired behavior.

General desired behavior example

This section shows some example general behaviors in the subsections on documents and messages:

Documents	**Messages**
Documents have the associated standard classification for the user group. For most users, this could be General, and for special groups, such as HR, Legal, or Finance, this could be Confidential.	All messages have a standard classification for the associated user group. For most users, this could be General, and for special groups, such as HR, Legal, or Finance, this could be Confidential.
The user must explicitly select a classification before the document is saved or printed.	Users should be asked to confirm the classification before the message is sent.
Users should be notified that they need to change the classification to the recommended label, determined by an automatic classification approach, which checks the document for specific patterns or keywords.	Users can change the classification before sending the message.

Now that we have worked through an example classification schema, including the labels, rules, and visual markings, we can discuss the different roles to process and manage data.

Defining the data-processing roles

It is important to set up the necessary administrative roles and rights. The following table identifies the different data owner roles and their respective rights:

Role	Create	Modify/delete	Delegation	Read	Archive/recover
Owner	X	X	X	X	X
Manager			X		
Administrator					X
User		X		X	

The following tasks can be performed by each role:

- **Data owners**: The owner of the information is the original creator of the data, who can delegate the ownership or delegate the administration to a data manager. When a file is created, the owner should be able to assign a classification. For this reason, they must understand the classification scheme and the associated guidelines, or be supported in the classification process in order to be able to set the correct classification. Finally, the data owner decides on the access to the information.
- **Data managers**: The data manager is instructed by the data owner to manage the information. They manage the information in accordance with the data owner's instructions, taking into account the applicable guidelines and requirements.
- **Administrator**: An administrator represents a user who is responsible for ensuring and maintaining the integrity of the data. The administrator is not a data owner. The administrator role includes backing up and restoring data and managing records.
- **Users**: Includes anyone who has been granted access to data or a file. Users have only user privileges and nothing else.

Finally, after discussing the different roles needed to process and manage data, we'll look at a short description about the change of the data classification label in the upcoming section.

Change of classification

Reclassifying or changing the classification status of an item of information must be done by a user, or the system determines that the importance or risk profile of the object has changed. This effort is essential to ensure that the classification status remains current and valid. Content that is not classified manually may be adjusted automatically or based on the use of the data owner or data manager.

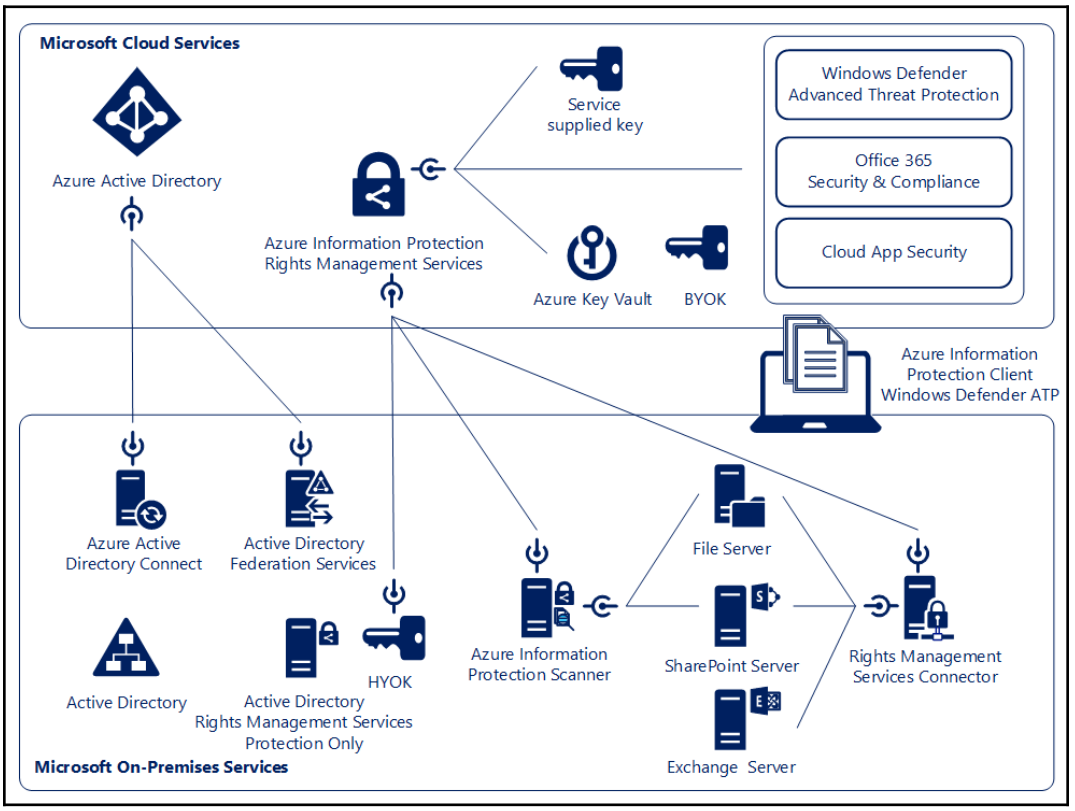
In this section, we have worked through a general overview of data classification, which we will use in `Chapter 13`, *Identifying and Detecting Sensitive Data,* and `Chapter 15`, *Configuring Azure Information Protection Solutions*. To provide an overview of the critical technology that we will use, we work through an introduction to **Azure Information Protection** (**AIP**) in the next section.

Azure Information Protection (AIP) overview

Microsoft has developed a complete data-classification and data-protection solution to meet the current and future needs of its customers and partners. AIP was designed based on the following form factors:

- Classify your data based on sensitivity
- Protect your data at all times
- Add visibility and control for users and administrators
- Support a more secure way of collaborating
- Easy-to-use toolset
- High deployment and management capabilities

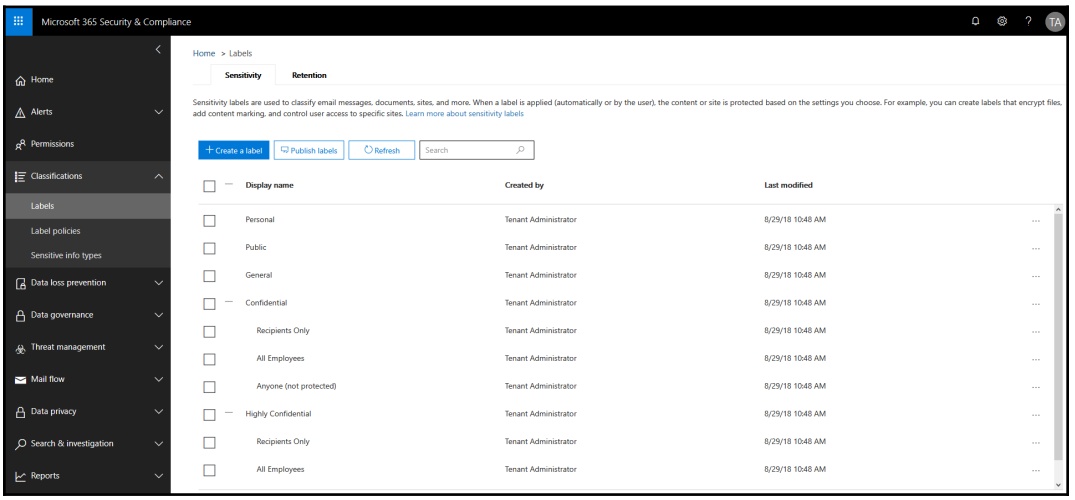
The following diagram shows the high-level architecture of AIP and the related security services. In the coming chapters, we will do a deep dive into each component and build a sample solution configuration:



The following benefits are provided by AIP:

- **Policy settings**: The administrators of AIP are provided with a set of the most common default labels, which can be modified to fit your requirements and needs

The following example shows the new Unified Labeling capabilities through the **Microsoft 365 Security & Compliance** center, which merges the Office 365 and the Azure Information Protection labeling system:



- **Classification**: Data can be classified based on content, context, and source, as well as automatically or manually by users, or in a combination.
    - **Example**: Combination of manual and automatic classification is recommended. It is shown to the user as follows:

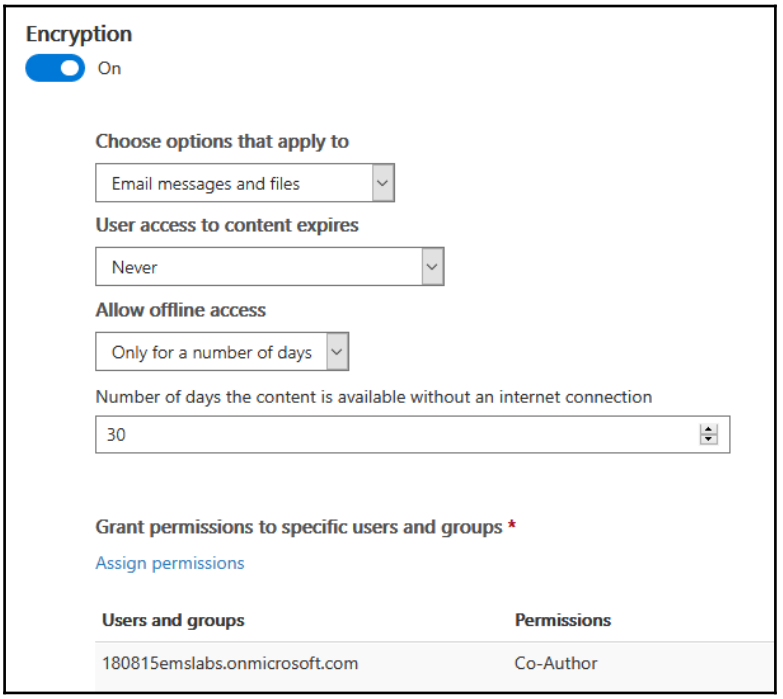
- **Labeling**: Labels will be added as metadata within a document. The labels are included in clear text so that other systems can read the values, for example the own Cloud App Security solution, which is a Cloud Access Security broker from Microsoft.

The following example shows the metadata of a classified document:

```
MSIP_Label_193d509a-931b-427b-bd10-45fa5a9ba362_Enabled True
 MSIP_Label_193d509a-931b-427b-bd10-45fa5a9ba362_SiteId
44b1e89f-e859-4a9b-8168-3438aea2529a
 MSIP_Label_193d509a-931b-427b-bd10-45fa5a9ba362_Owner
jochen.nickel@emslabs.ch
 MSIP_Label_193d509a-931b-427b-bd10-45fa5a9ba362_SetDate
```

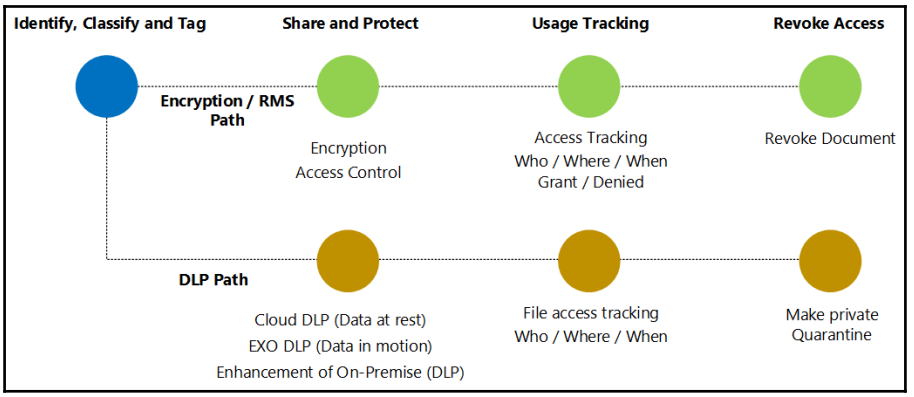
```
2018-11-12T14:53:49.6671922Z
 MSIP_Label_193d509a-931b-427b-bd10-45fa5a9ba362_Name General
 MSIP_Label_193d509a-931b-427b-bd10-45fa5a9ba362_Application
Microsoft Azure Information Protection
 MSIP_Label_193d509a-931b-427b-
bd10-45fa5a9ba362_Extended_MSFT_Method Manual
 Sensitivity General
```

- **Protection**: The encryption of documents and the authentication requirements are solved by the existing Azure Rights Management services or Active Directory Rights Management services, which allow you to define fine-grained permissions on the protected information.
    - **Example**: Here, you can see custom protection settings in the Microsoft 365 Security & Compliance Center:



- **Monitoring, tracking, and logging**: Users can track activities on their shared information and can revoke access immediately. Administrators are also able to monitor and log information about the administrative tasks, classification, and protection events.
    - **Example**: **Revoke access** to a document:

The following diagram shows the main interactions on the main paths to protect information through its life cycle:



We can also see that the AIP solution is a part of the Microsoft Cyber Security Architecture (information protection). You can learn more about the cyber security architecture of Microsoft at `https://bit.ly/2Sh1zN2`.

AIP provides a suitable framework for data classification. There are still many limitations in the product. With other vendors, such as TITUS, you can fill the gaps and provide a suitable solution. You can find more information at `https://bit.ly/2Gbu0Wq`.

Now that we have seen a small starter on AIP, in the coming chapters, we will do a deep dive into this technology.

Summary

In this chapter, we learned why you need to have a security culture in an organization, and how it relates to and is supported by an information protection strategy. Then, we discussed what data classification means and why it builds the foundation of every data security solution. During the general overview of data classification, we looked at the relevance of a data classification scheme and related policies. Finally, we saw a Microsoft AIP solution overview, which builds the technology base for the information-protection solution we'll create in the coming chapters. Using the knowledge you gained in this chapter, you'll be able to define your own classification schemes, rules, and policies. Furthermore, you know which areas are critical in a security culture and you can include the appropriate tasks into your projects.

In the next chapter, we will dive into our first practical experience with AIP: identifying and detecting sensitive data.

13
Identifying and Detecting Sensitive Data

Identifying and detecting sensitive data is a very important process inside an information-protection solution. You need to be able to identify sensitive information within your environment using suitable matching criteria to provide the related results for appropriate classification and protection controls. This is quite important to give the right access to the right person and to improve your security standards. Microsoft has invested in this field and provides different solutions that work natively together. The solutions provide capabilities for data in motion, in transit, and at rest. In this chapter, we'll give a practical overview of the needed information that can be used for your organization or customers:

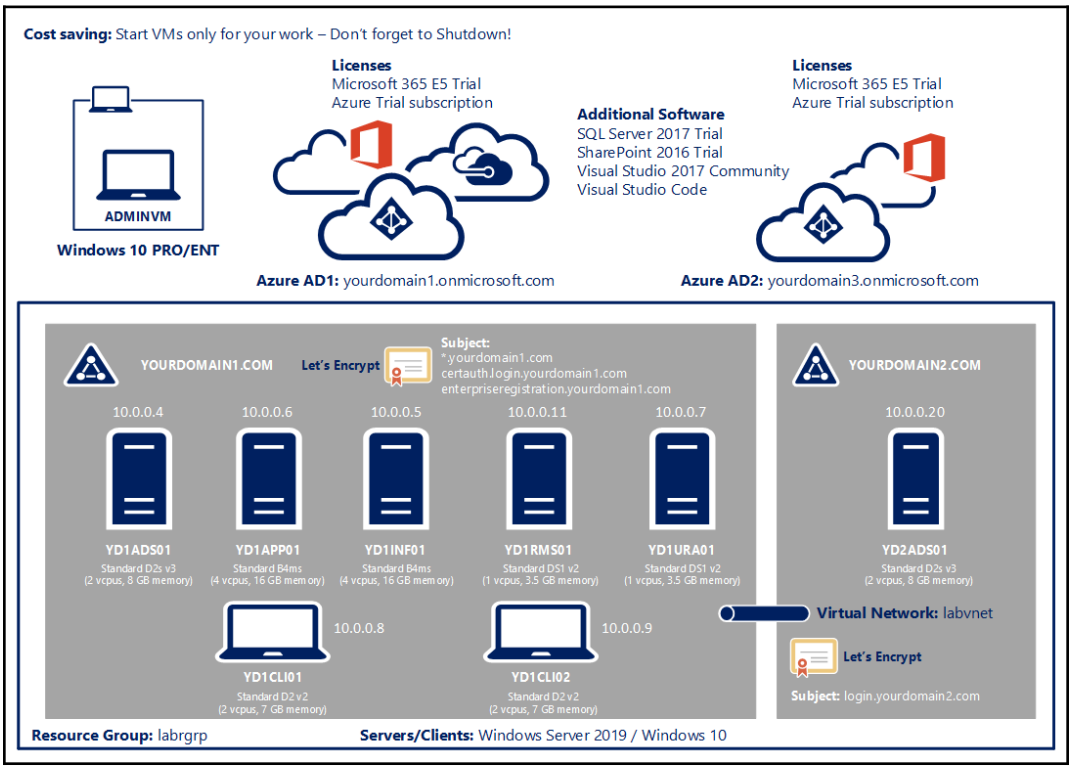
- Extending your lab environment
- Understanding and using AIP capabilities for data in motion
- Understanding and using AIP capabilities for data at rest

In the first part of the chapter, we will extend your lab environment.

Extending your lab environment

First, we need to extend your lab environment to test the different features and give you the chance to extend the scenarios as you work through the different chapters in this book.

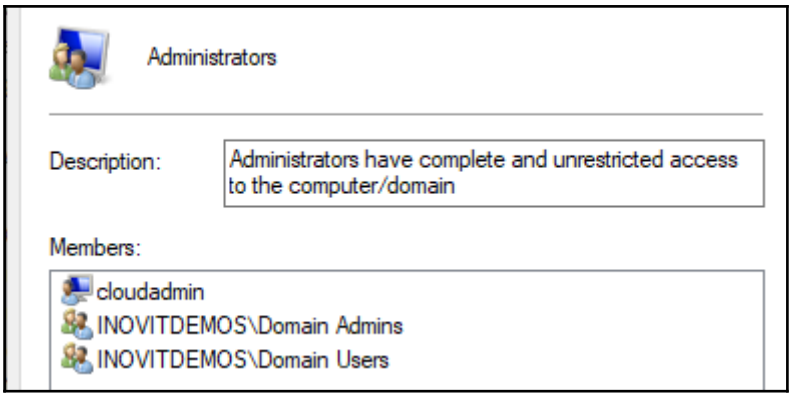
We will add two Windows 10 test clients and the **YD1INF01** server. Use the following diagram to get the correct sizing, and the virtual machine configuration and domain membership for the virtual machines:



Lab environment overview

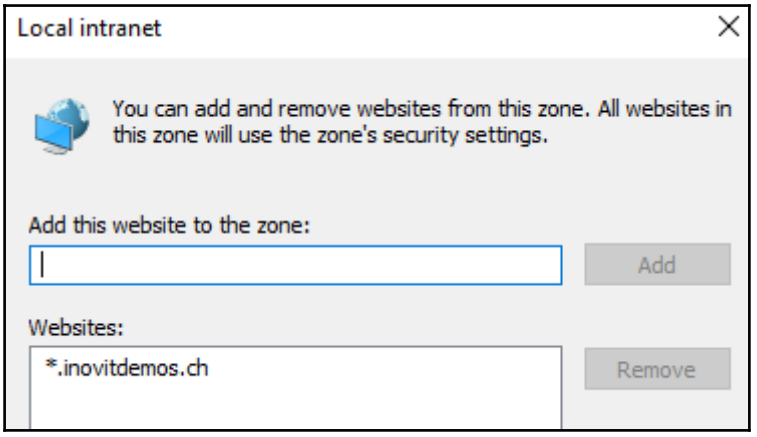
The Windows 10 clients need to be joined to **YOURDOMAIN1.COM** and you need to copy the example data files from the code package to any directory on the two clients:

1. Add the **Domain Users** in the local **Administrator** group to provide easy access to the virtual machines:



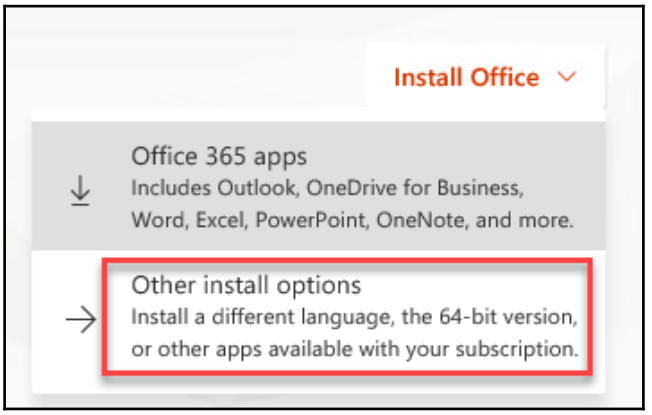
User access to VM

2. To use single-sign-on, add `*.yourdomain1.com` to your **Local intranet** zone in your Internet Explorer configuration:



IE Local Intranet zone configuration

3. Install **Office Suite** and the **Azure Information Protection** client on both virtual machines.
4. Open `https://portal.office.com` with your **Office 365 licensed admin user**:

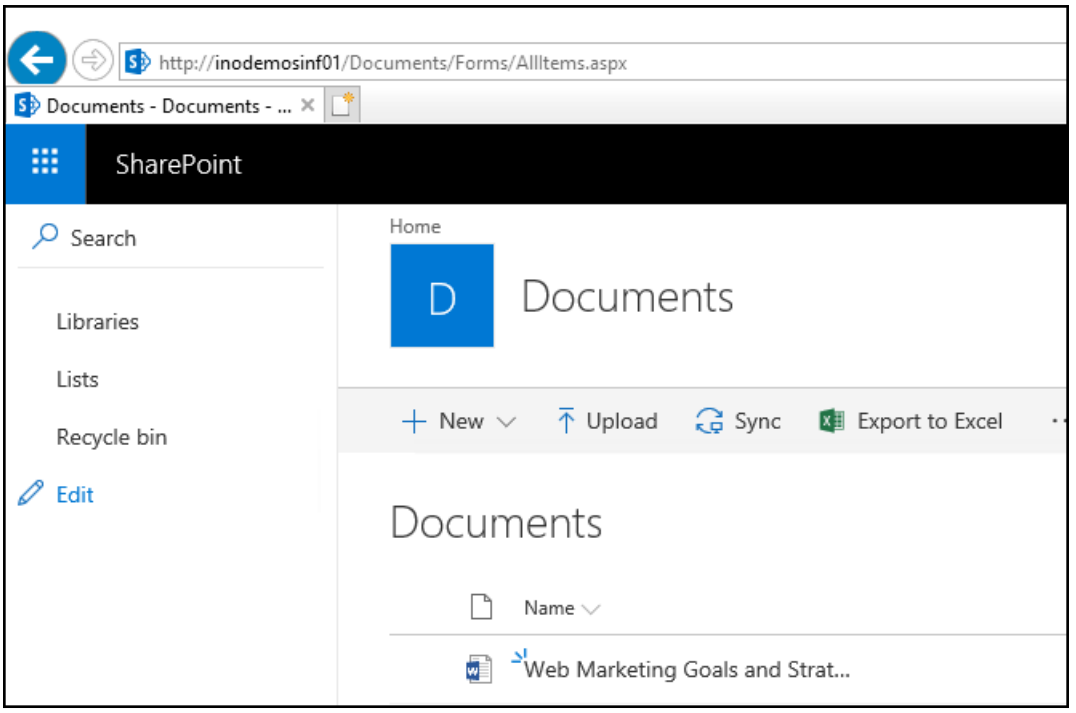


Installing the 64-bit version of Office

5. Install the 64-bit Version of Office on the client.
6. Download and install the Azure Information Protection client on the client. You will find the installer under `https://aka.ms/aipclient`.
7. Choose the `AzInfoProtection_PREVIEW_1.45.32.0.exe` binary and install it without a demo policy.

Additionally, we will install a SharePoint single farm on the **YD1INF01** server for testing the discovery capabilities on-premises:

1. Visit `https://bit.ly/2QFfBZ0` to install **SharePoint**.
2. The expected result is that we have a demo **SharePoint** infrastructure with a simple **Documents** library:



SharePoint library for test files

We configure an example file structure on the **YD1INF01** server to discover sensitive data on traditional file server solutions:

```
New-Item -ItemType directory -Path 'C:\Shares\Marketing'
New-Item -ItemType directory -Path 'C:\Shares\Finance'
New-Item -ItemType directory -Path 'C:\Shares\Production'
New-Item -ItemType directory -Path 'C:\Shares\Sales'
New-Item -ItemType directory -Path 'C:\Shares\Executives'
New-Item -ItemType directory -Path 'C:\Shares\HumanResources'
New-Item -ItemType directory -Path 'C:\Shares\ResearchDevelopment'
New-SmbShare -Name Marketing -Path 'C:\Shares\Marketing' -FullAccess
Everyone
New-SmbShare -Name Production -Path 'C:\Shares\Production' -FullAccess
Everyone
New-SmbShare -Name Sales -Path 'C:\Shares\Sales' -FullAccess Everyone
New-SmbShare -Name HumanResources -Path 'C:\Shares\HumanResources' -
FullAccess Everyone
New-SmbShare -Name ResearchDevelopment -Path
'C:\Shares\ResearchDevelopment' -FullAccess Everyone
New-SmbShare -Name Finance -Path 'C:\Shares\Finance' -FullAccess
```

```
Everyone
New-SmbShare -Name Executives -Path 'C:\Shares\Executives' -FullAccess
Everyone
```

We'll copy data during the relevant configuration steps in the *Understanding and using AIP capabilities for data in motion* section. After completing the extension of your lab environment, let's identify and detect data in motion.

Understanding and using AIP capabilities for data in motion

The meaning of data in motion or transit is that information is actively moving from one location to another inside your environment or outside through the internet. Typically, the process starts with the creation of a document, presentation, or spreadsheet, or the export of information from a source, such as the HR, CRM, or another system. The best option is to include the classification and protection process directly in these processes. Otherwise, you need to be able to detect and identify sensitive data from scratch on client computers and servers. Another important thing is that you know where your sensitive, protected, or not-protected information is exchanged and stored. Let's say you need a solution that's able to monitor the sensitive data on client computers, on-premises, and in the cloud. Microsoft follows this strategy and provides the following technologies to fulfill these requirements to actively monitor sensitive data in motion:

- **Azure Information Protection**: Classification and protection monitoring including document tracking.
- **Windows Defender ATP**: Monitoring classification and protection activities on client computers.
- **Cloud App Security**: Active monitoring of your enabled cloud ecosystem, including data-leakage prevention, classification, and protection capabilities.
- **Office 365**: Monitoring, data-leakage prevention, classification, and protection capabilities inside the Office 365 services.

Let's dive a bit deeper into the different functionalities of these technologies.

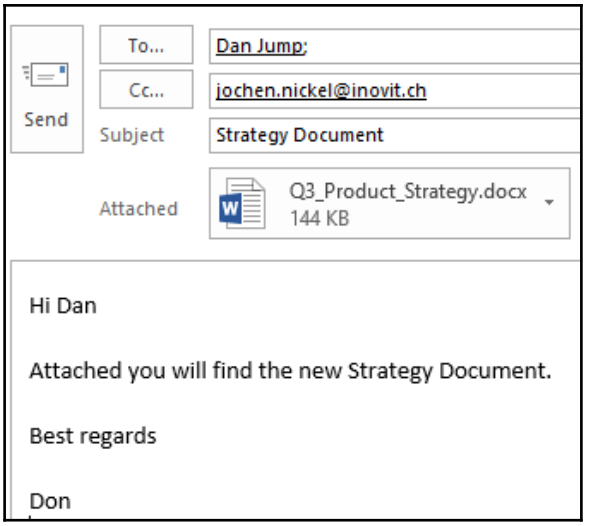
Scenario 1 – Usage of Azure Information Protection

The Azure Information Protection client offers you the capability to classify and protect information in several formats. In our example, we'll use a manual classification that also applies protection to the document. We use the `Q3_Product_Strategy.docx` document from the code package for this example:

1. Log on to one of the test clients with `don.hall@yourdomain1.com`. Don Hall is a Strategy Consultant.
2. Open the `Q3_Product_Strategy.docx` document.
3. Because of the sensitive content, Don classifies the document as **Confidential | All Employees** to classify and protect the information against people outside his organization, which is shown in the following image:

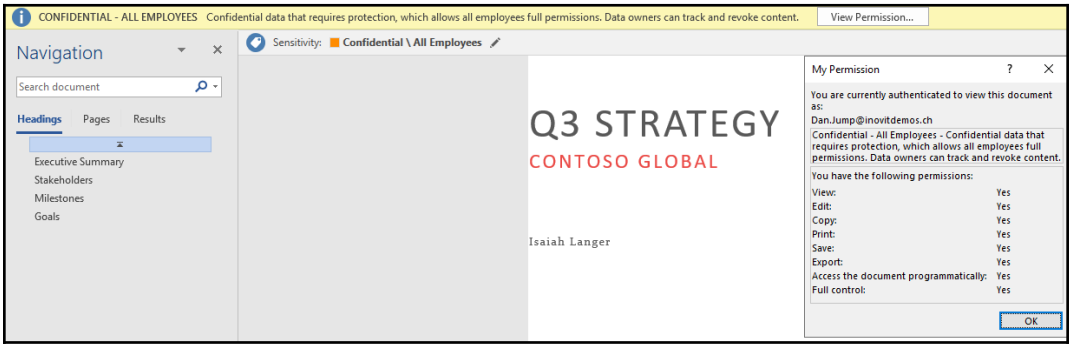
All Employees AIP label

4. Save the file and send it to any of your personal email addresses and to `dan.jump@yourdomain1.com`, the CEO of our demo (be sure that he has a Microsoft 365 E5 license assigned):



Test mail including external contact

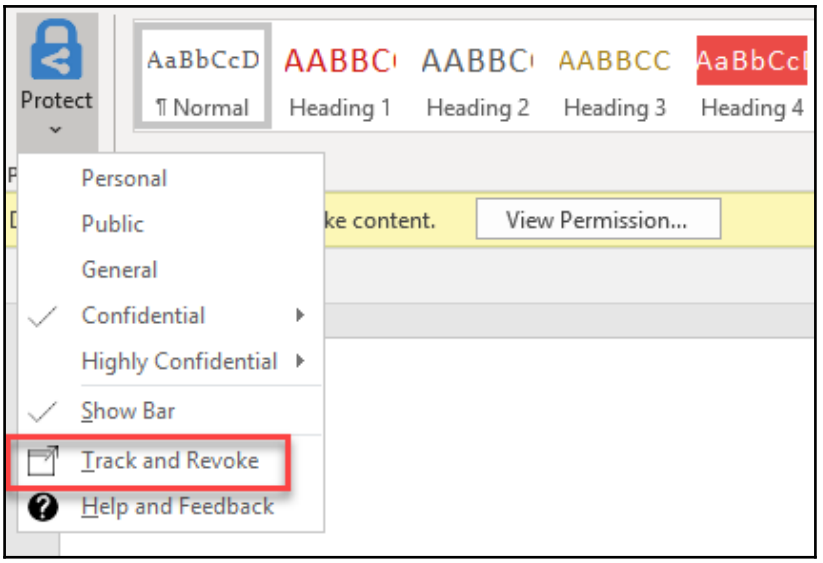
5. Try to open the email attachment on your private account—you won't get access!
6. Log on to the second Windows 10 client with the `dan.jump@yourdomain1.com` user.
7. Open Outlook, open the attachment, and click **View Permission...**:



Protected document with associated rights

Now that we've worked on a document, and classified and protected it, including exchanging the information with an internal employee and an external account, we can view the monitoring capabilities of Azure Information Protection by following the steps listed here:

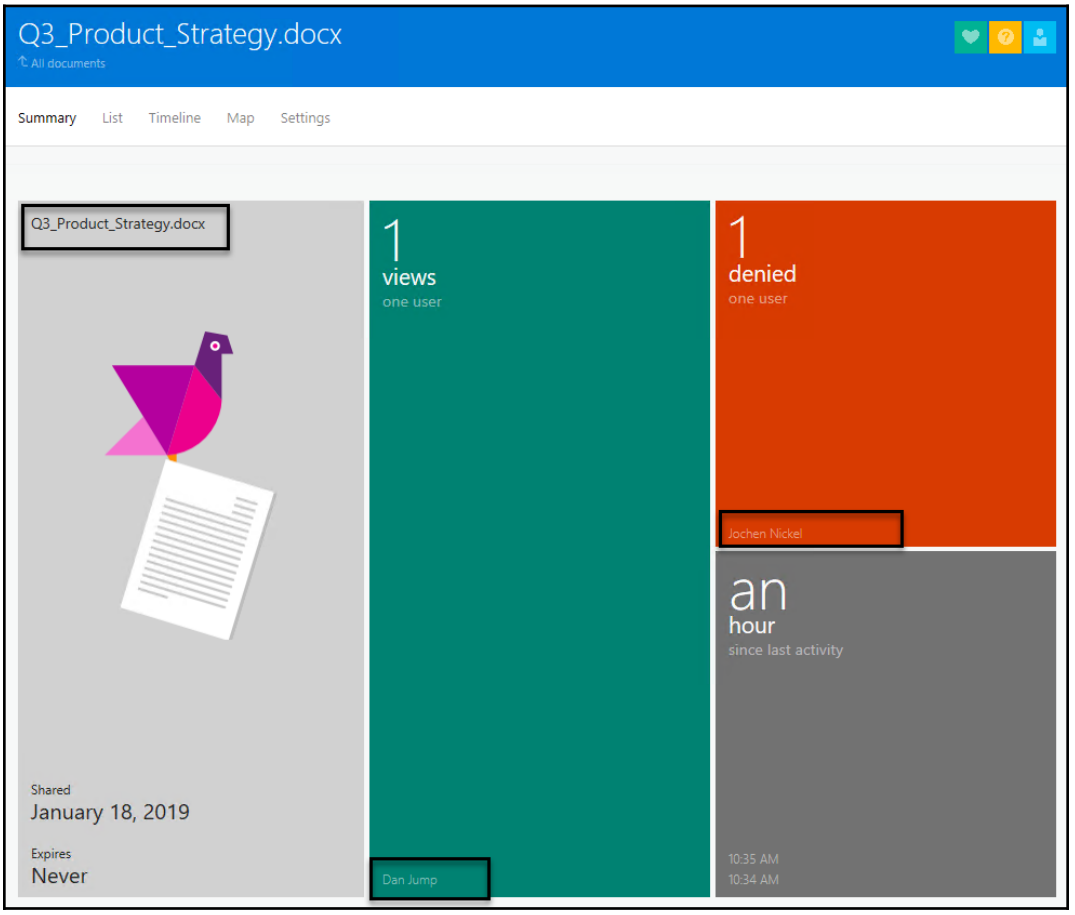
1. Go back to the Don Hall client and open the protected document, `Q3_Product_Strategy.docx`, again.
2. Click the Azure Information Protection **Protect** button and choose **Track and Revoke** as shown in the image here:



RMS Track and Revoke feature

3. You will be redirected to the Azure RMS tracking website: `https://track.azurerms.com/`.
4. Log in with `don.hall@yourdomain1.com`.

5. You'll get all the information about your protected document as shown in the following screenshot:

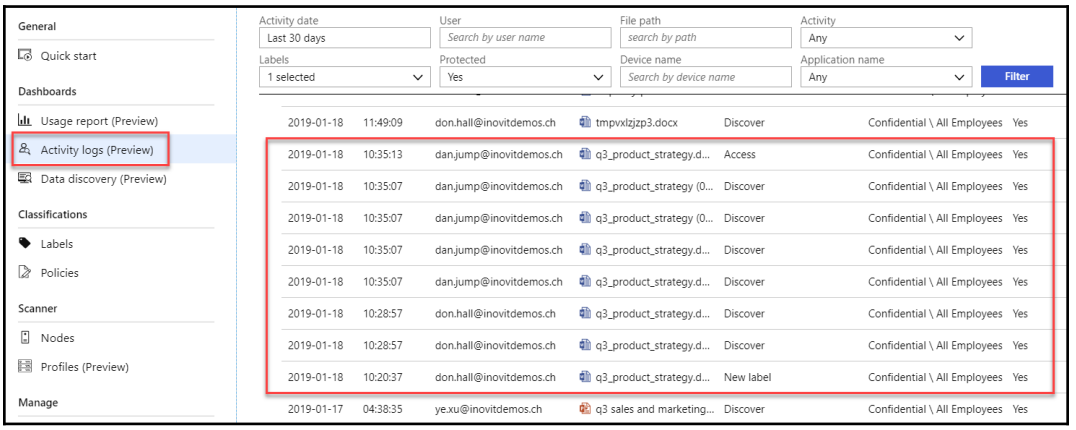


Tracking report on the protected document

6. View all the capabilities you have as a user under **Summary** | **List** | **Timeline** | **Map and Settings**.

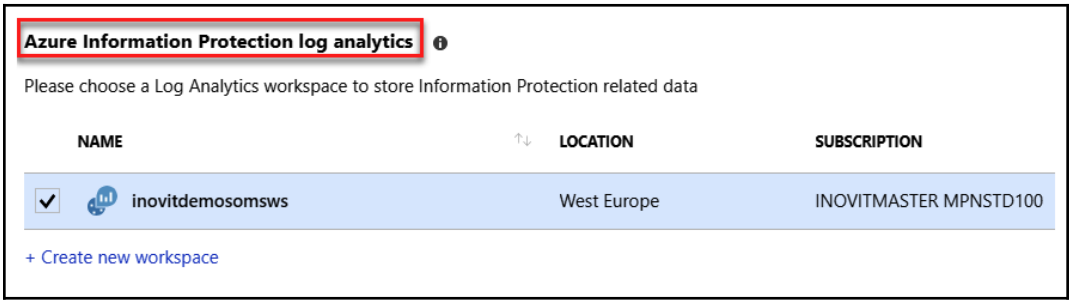
Next, we'll look into the administrative portal blade of **Azure Information Protection**:

1. Under the **Activity Logs** section, you'll be able to view all the activities on `Q3_Product_Strategy.docx`:



Activity logs of AIP operations

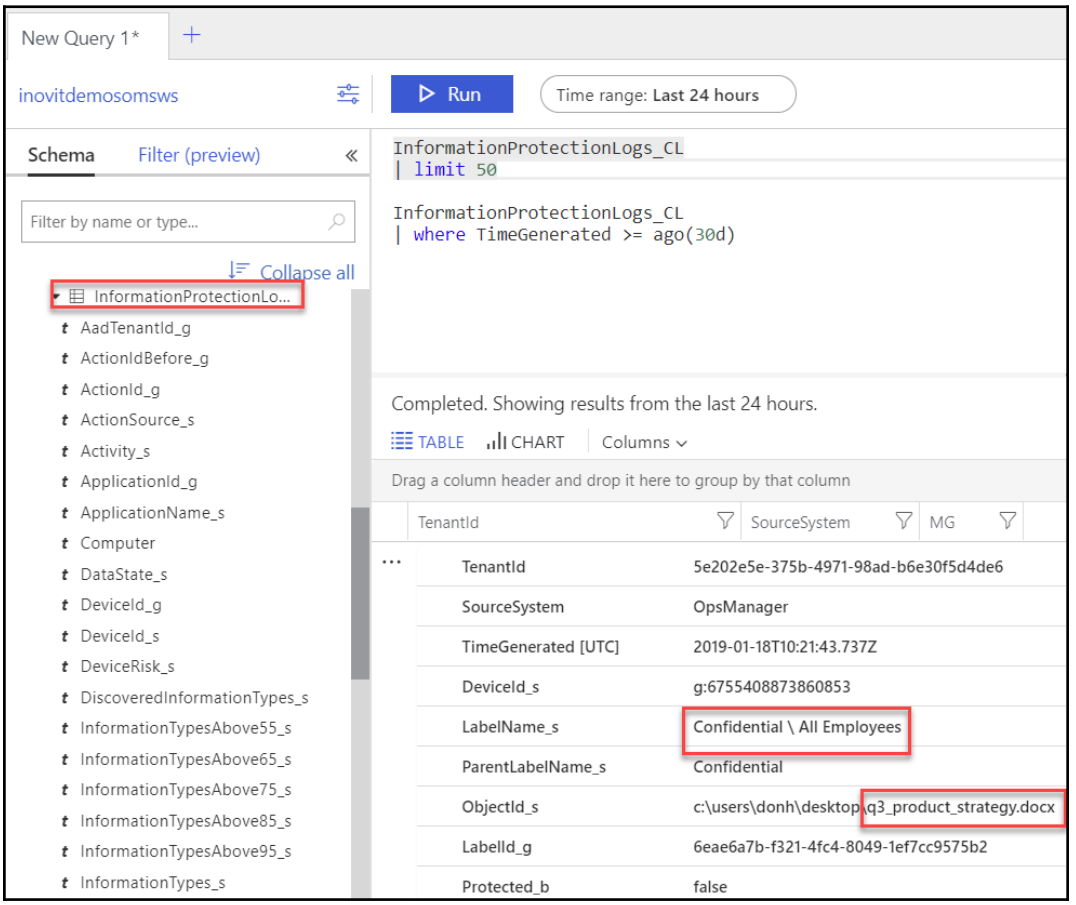
As we've already configured **Azure Log Analytics** in `Chapter 5`, *Configuring and Managing Identity Protection*, we can use the same **Log Analytics** workspace for our Azure Information Protection data under the **Configure analytics (Preview)** section. Ensure that the **Enable the document content matches feature...** checkbox is enabled:



Log analytics store

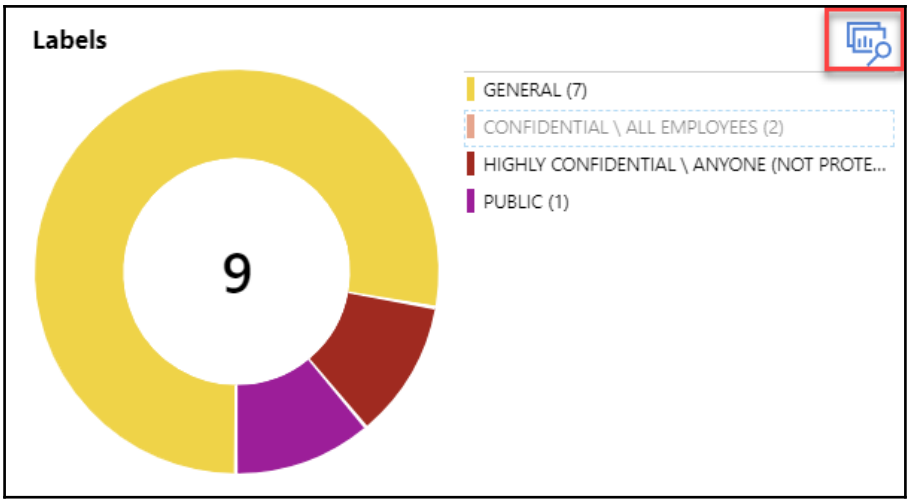
2. Under **Activity logs (Preview)**, you will find the **Log Analytics** button—click it!

3. We'll be shown more detailed information about our classification and protection task with `don.hall@yourdomain1.com`:



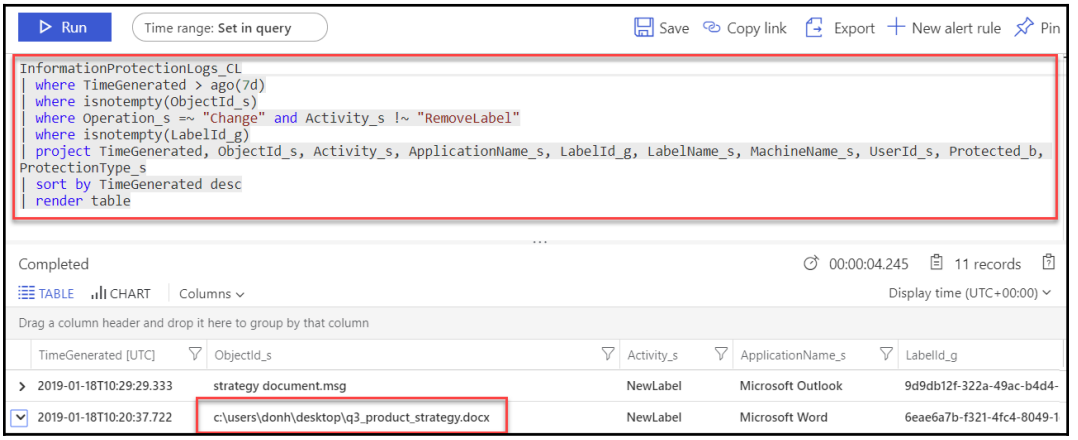
Detailed log in Azure Log Analytics

4. You also can drill down directly over the **Usage report (preview)** section, which is demonstrated in the following image:



Labels usage overview

5. Click the **Log Analytics** icon to view the generated query and the result:



Specific log results of the document

To identify and classify more information, we'll use several methods such as keywords, or prepared sets of sensitive information, in `Chapter 15`, *Configuring Azure Information Protection Solutions*. You can come back to the monitoring part and see the results.

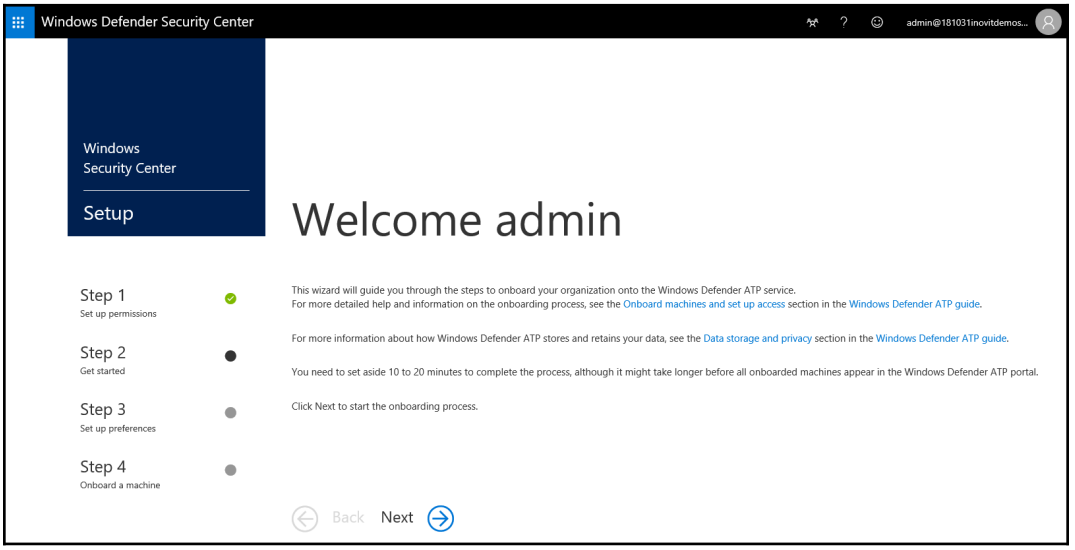
Now that we've seen the standard capabilities of Azure Information Protection for users and administrators to discover sensitive information, we can view the Windows Defender ATP features.

Scenario 2 – Monitoring with Windows Defender ATP

Windows Defender Advanced Threat Protection is basically designed to prevent, detect, investigate, and respond to advanced threats. We can also use it to detect and identify sensitive information, especially on client systems.

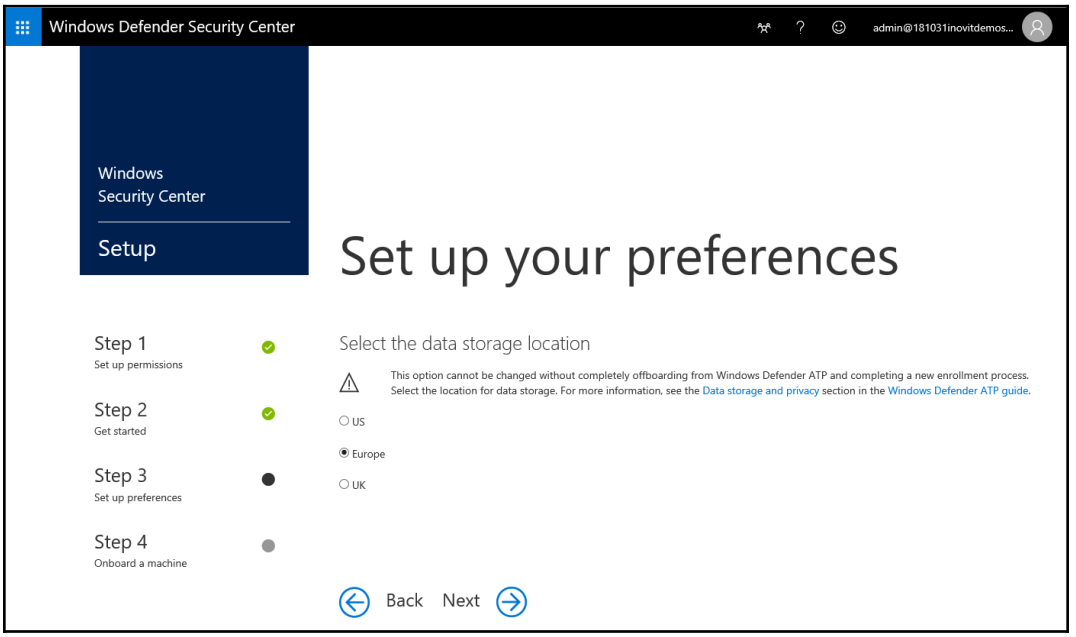
To work with this feature, we need to configure it and onboard the two test clients:

1. Visit `https://securitycenter.windows.com/` and log in as **global administrator** to start the configuration as shown in the image here:
    - Click **Next**:



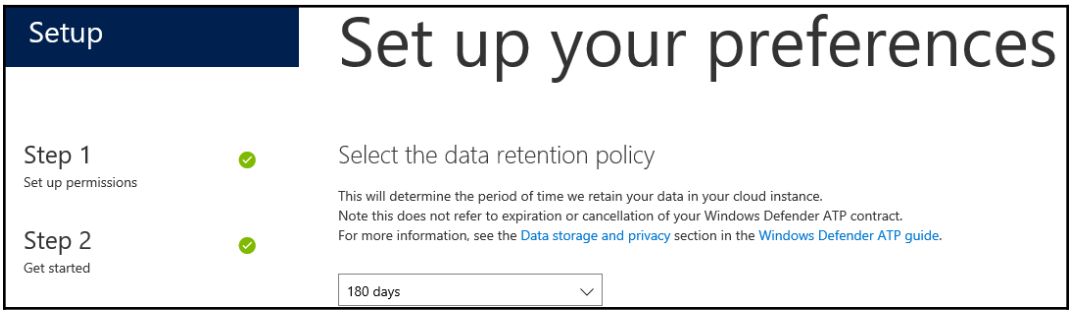
Windows Security Center portal

- Choose your **storage location** based on your needs:



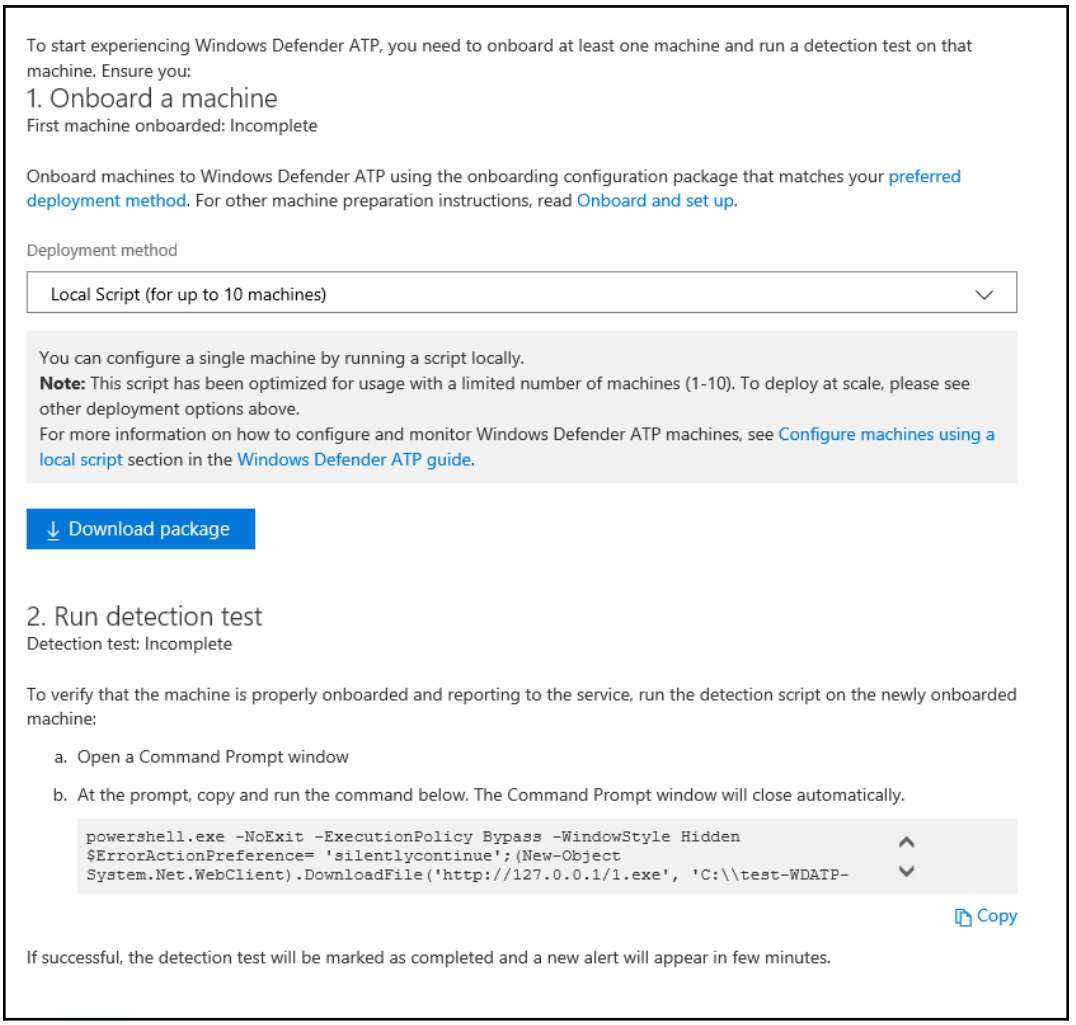
Data storage configuration

- Select the data retention policy.



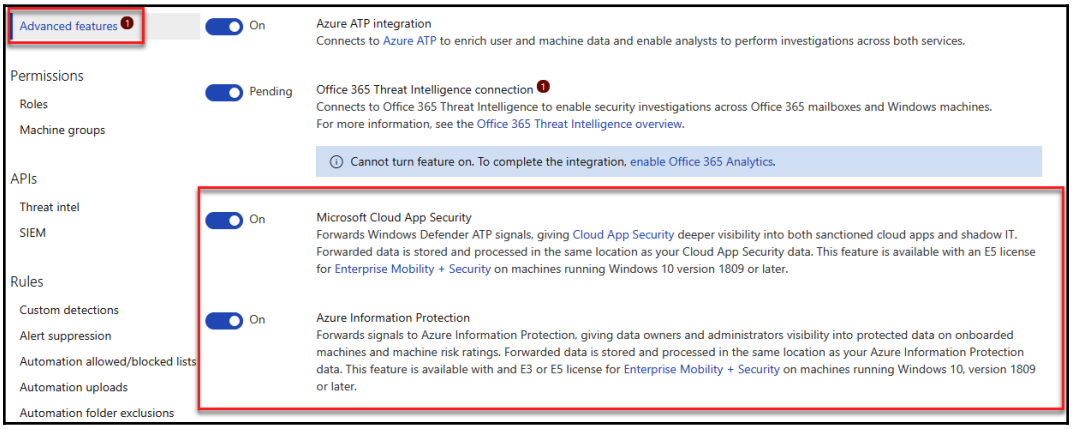
Retention time configuration

2. Select your organization size, the industry, and the preview experience.
3. Onboard the two test clients:
    - Download the package and install it.
    - Run the detection test script on both clients:



Onboarding the test script

4. Activate the Advanced features, including Azure Information Protection and Microsoft Cloud App Security, under **Settings** | **General** | **Advanced Features**:



Connected services configuration

5. The two demo clients will appear under the machine list:

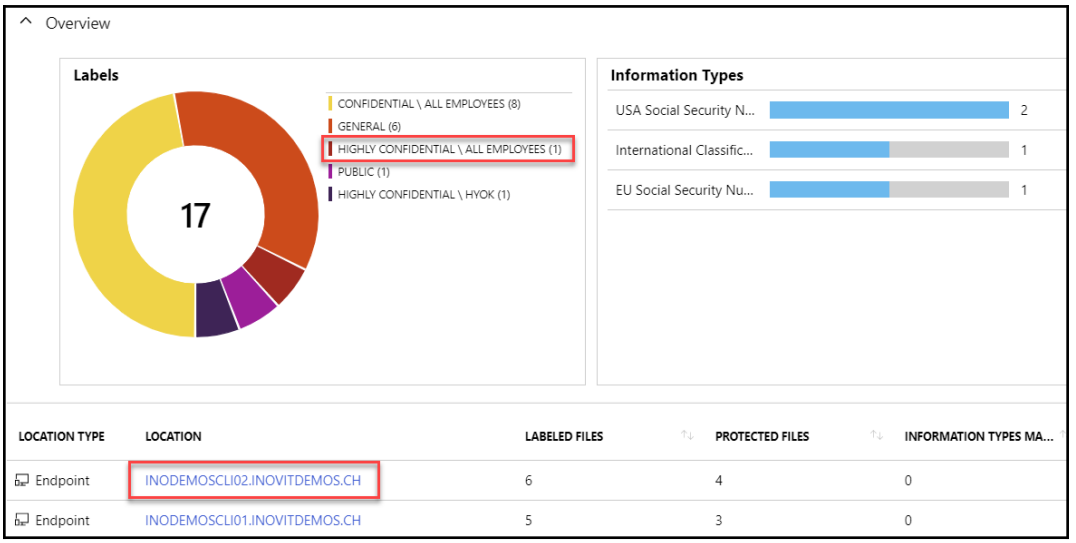
Machine list

After the successful activation and onboarding of the two client machines to **Windows Defender ATP**, we're ready to test the features for our sensitive data discovery:

1. Log on to one of the test clients with `dan.jump@yourdomain1.com`. Dan Jump is the CEO.
2. Open the `Employee Details.xlsx` spreadsheet.
3. Classify the document as **Highly Confidential** | **All Employees**.

In `Chapter 15`, *Configuring Azure Information Protection Solutions*, we'll provide the related classification and labels for the specific use cases. For now, it's just important to generate, modify, and use sensitive data.

4. Wait a few minutes to receive the results in the Data Discovery (Preview) section of Azure Information Protection:



Label report from the clients

5. Click on the client **Endpoint**:

Client endpoint information

Awesome! This integration between the different services will let you discover sensitive data-handling in your environment. With Windows Defender ATP, Azure ATP, and Azure Information Protection, you can associate the three most important elements:

- The identity, including the authentication information
- The data-handling associated to the identity itself
- The attacks against computers, servers your identity provider and at the end to your sensitive information

In the next section, we'll dive into the Cloud App Security feature set to identify sensitive data in your cloud ecosystem.

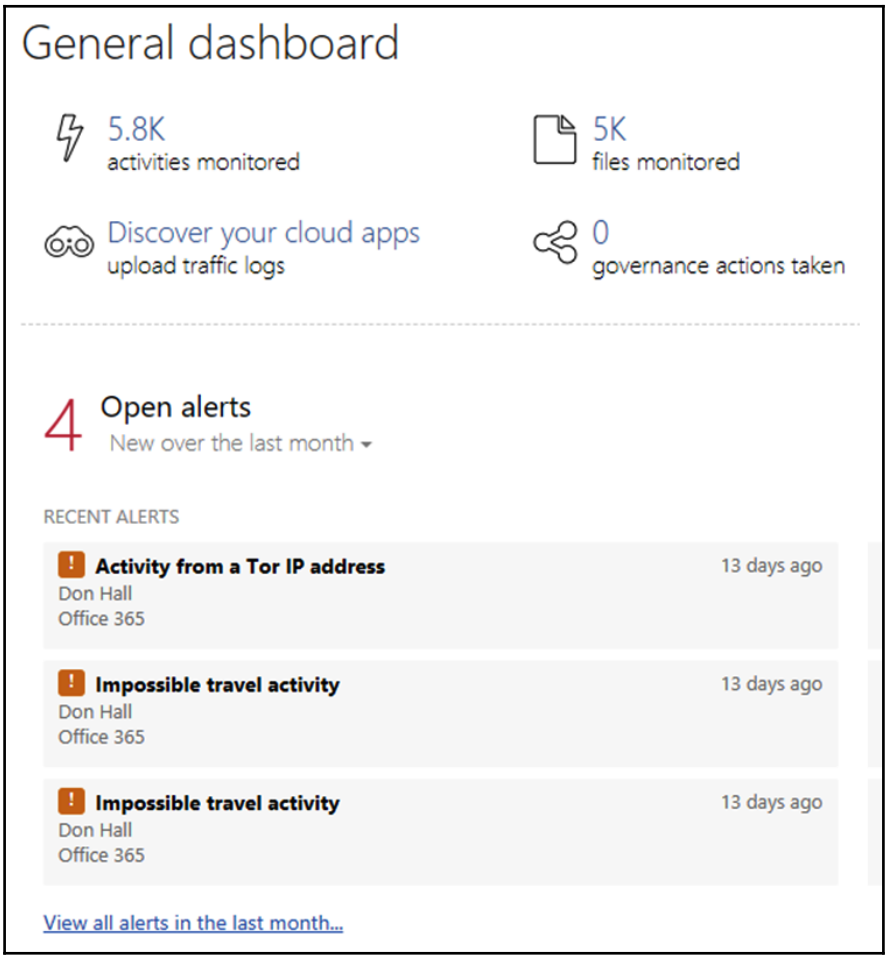
Scenario 3 – Identifying sensitive information in your cloud ecosystem

Microsoft Cloud App Security is a **Cloud Access Security Broker** (**CASB**) that gives you the needed insights into your cloud app and services ecosystem. With this service, you're able to control how your data travels, and you can identify and work against cyber threats. In this section, we'll take a closer look at the discovery of sensitive information inside our demo environment.

First, we need to enable Cloud App Security and connect our cloud apps to monitor our cloud ecosystem. We also need to configure the Azure Information Protection integration to combine the services.

Visit `https://portal.cloudappsecurity.com` and log in with the global administrator credentials.

If you're working through all the labs in this book, your **General dashboard** will look like the following:

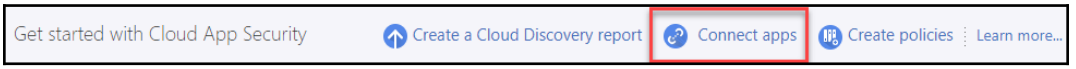


Cloud app security dashboard

With the following steps, we start the configuration:

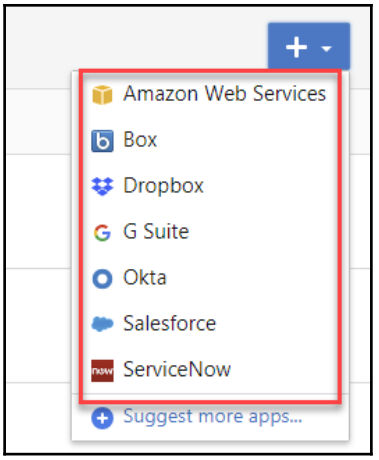
1. Connect the apps we configured in the previous chapters to the Cloud App Security framework:
    - Office 365
    - Salesforce
    - Dropbox
    - Microsoft Azure

2. This can be done through the getting started wizard—click **Connect apps**:



Connecting additional apps to cloud app security

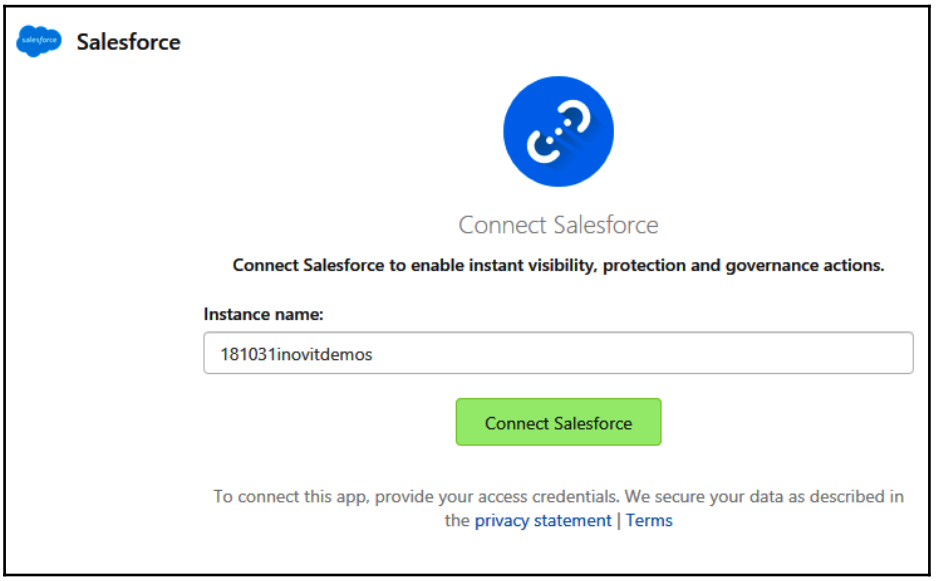
3. Connect these apps with the **+** dropdown.
4. Provide the **related administrative credentials** to connect the apps:



Cloud app connectors

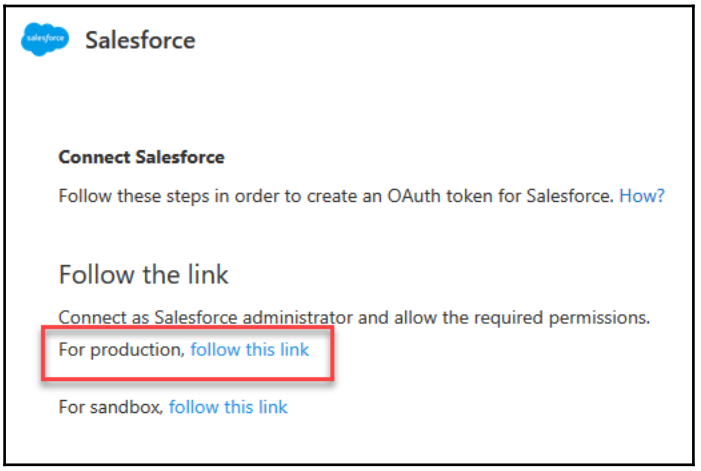
To give you an example, we'll demonstrate the process with **Salesforce**:

1. Provide your **Salesforce** instance name:



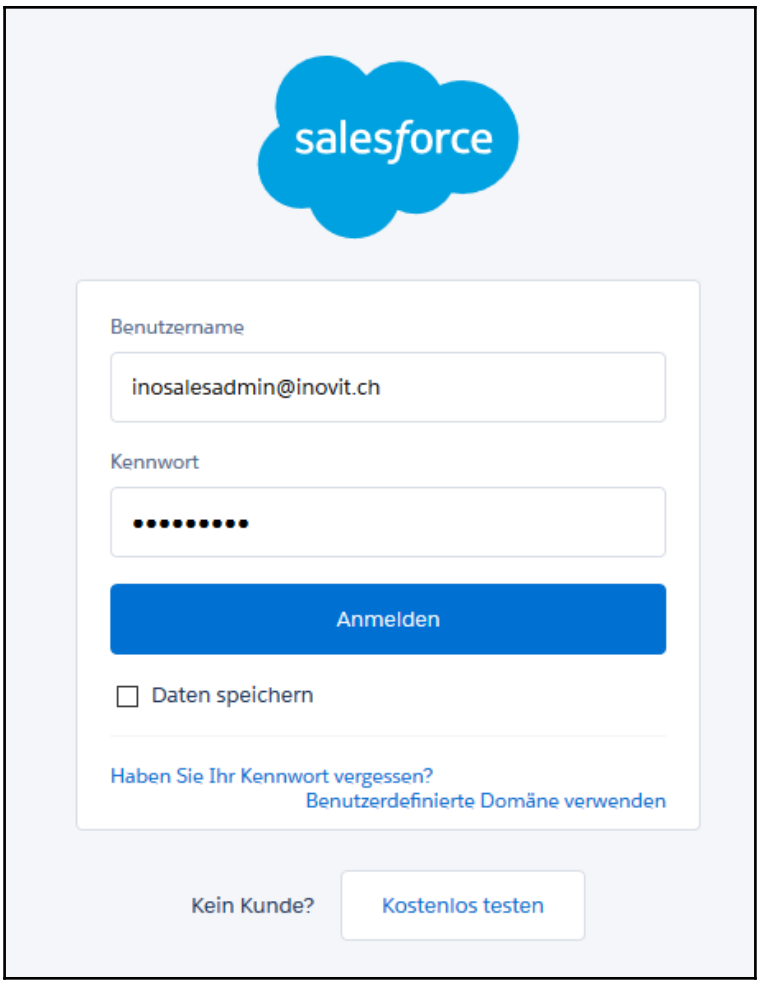
Connecting Salesforce

2. Follow the link to connect the app and allow the required permissions:



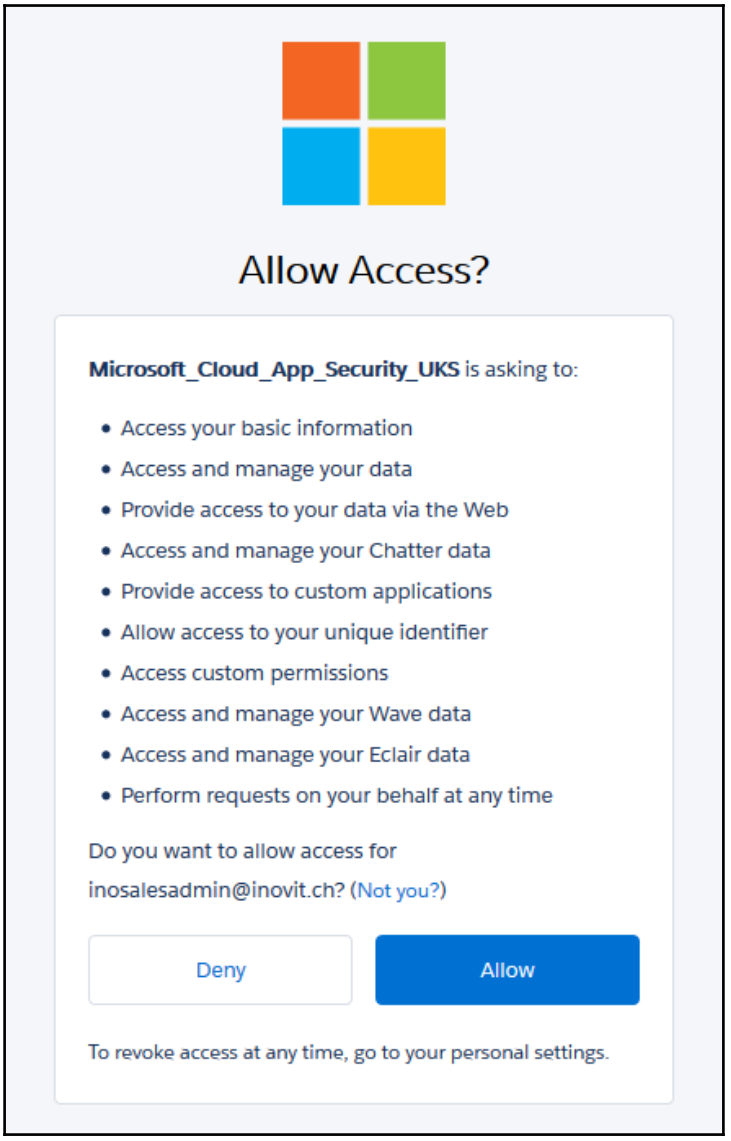
Permission assignment

3. Provide your administrative credentials:



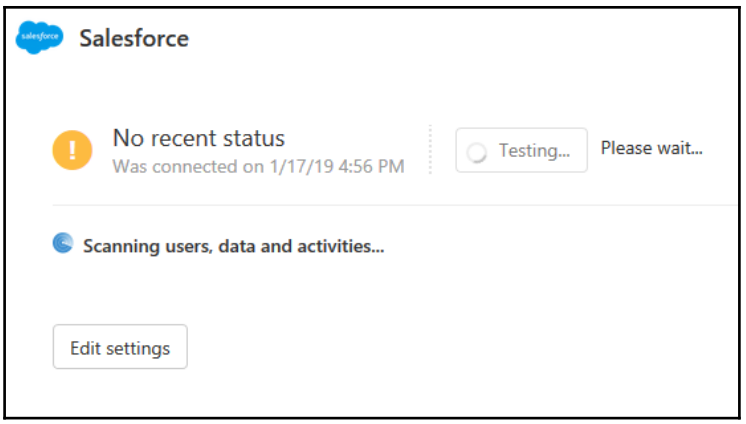
Providing admin credentials

4. You will be requested to consent—click **Allow**:



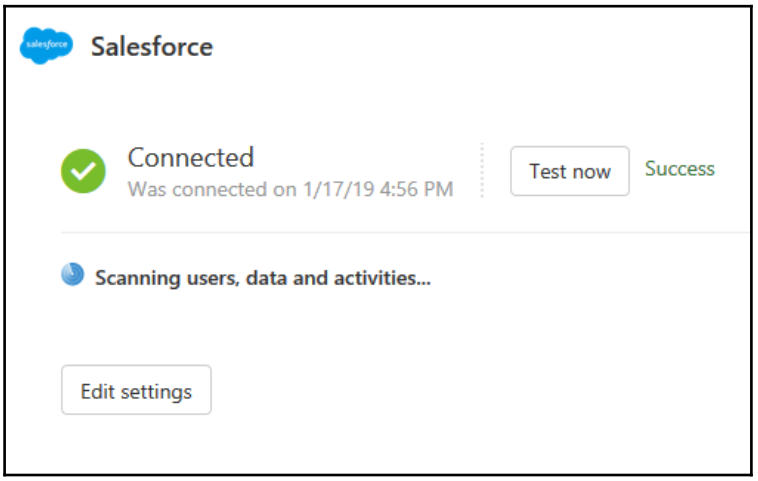
Checking and accepting OAuth consent

5. Cloud App Security will start scanning the users, data, and activities:



Cloud app security starting the discovery process

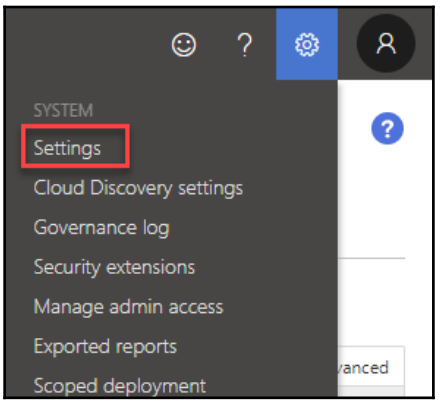
6. You will receive a **Connected** status:



Salesforce connection status

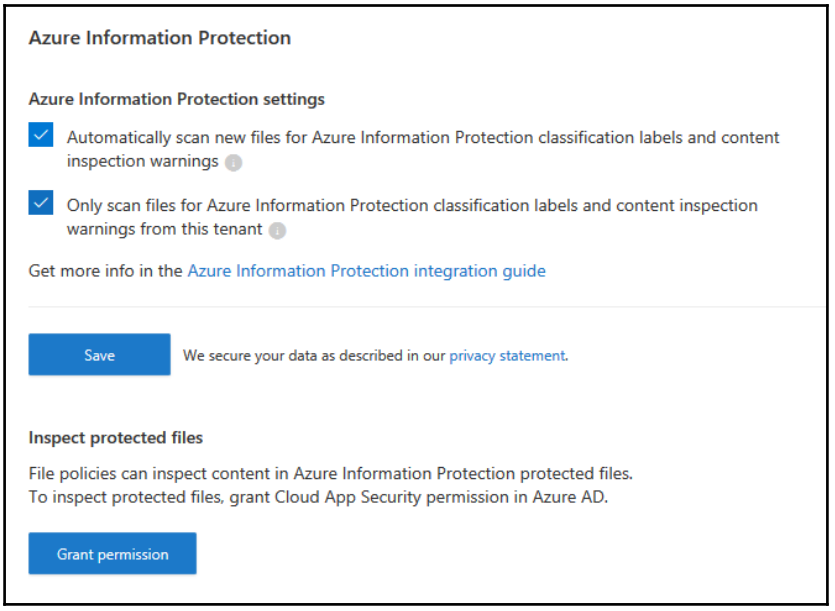
Use the wizards for every application that's connected to Cloud App Security.

7. Activate the Azure Information Protection integration in the **Settings** section:



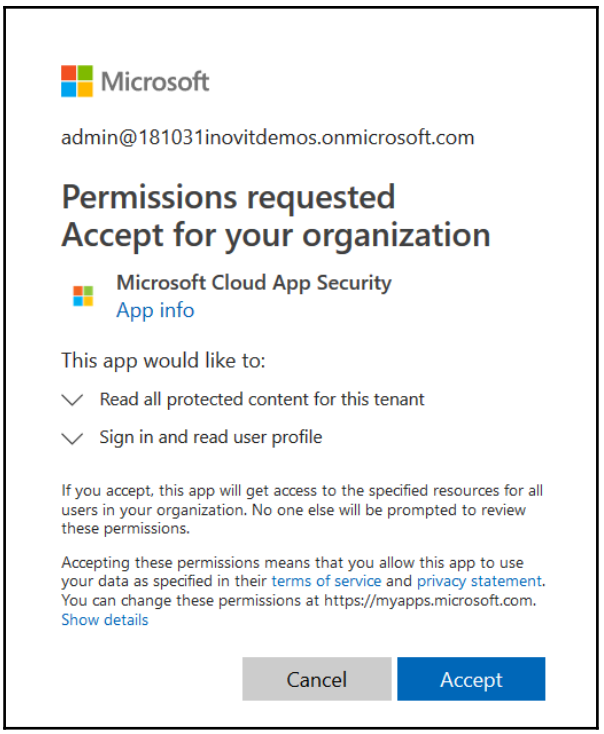
Configuring AIP integration

8. Navigate to **Information Protection** | **Azure Information Protection** and set active the following two settings:



AIP settings in Cloud App Security

9. Grant permission to Cloud App Security to inspect protected files:



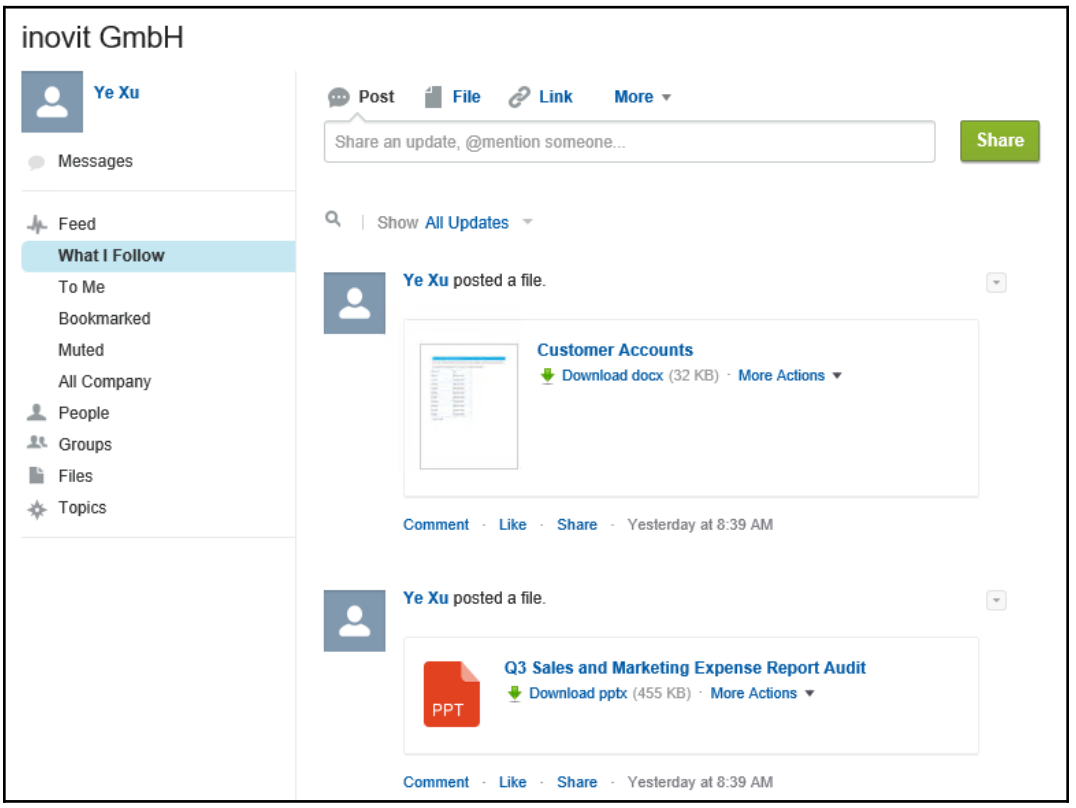
OAuth consent to assign permissions

10. Click **Accept**.

Now, that we have connected our apps to Cloud App Security and configured the Azure Information protection, we can start, to work through our next tasks. With the following steps we start the configuration:

1. Log in to one of the test clients with `ye.xu@yourdomain1.com`. Ye Xu works in Sales.
2. Log in to your Access Panel UI: `https://myapps.microsoft.com`.
3. Click **Salesforce**.

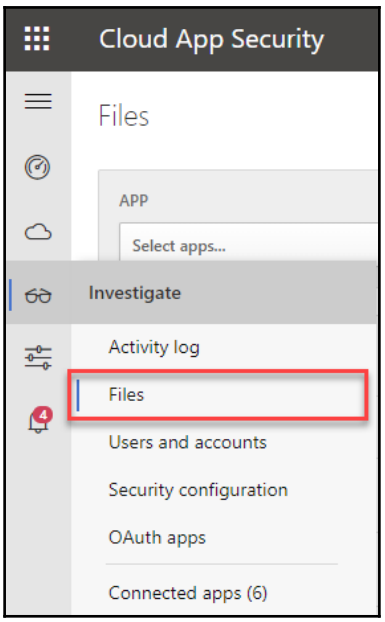
4. Post the following files from the code package:
    - Q3 Sales and Marketing Expense Report Audit
    - Employee Details
    - de-drv
    - Superstore Sales
    - Customer Accounts:



Salesforce user view and data usage behavior

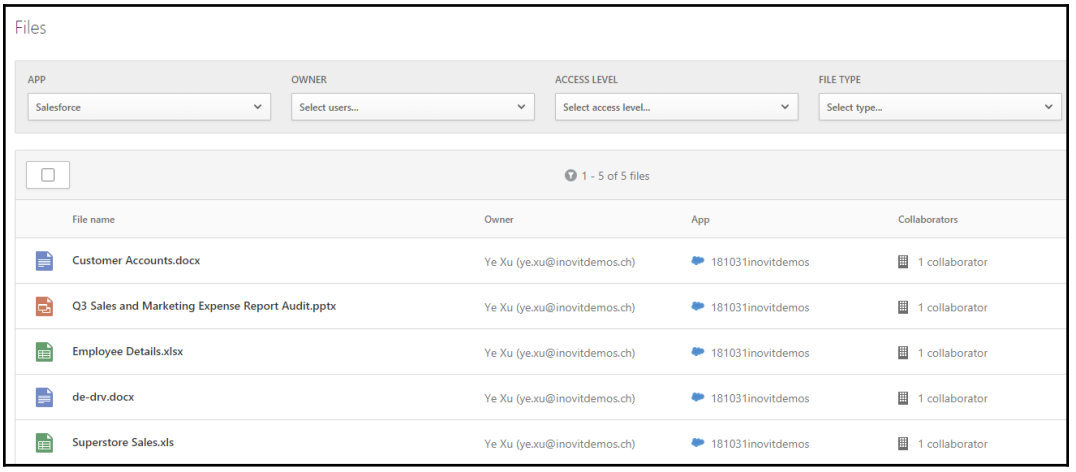
5. Jump back to the Cloud App Security portal (`https://portal.cloudappsecurity.com`) as a global administrator.

6. Navigate to **Investigate** | **Files**:



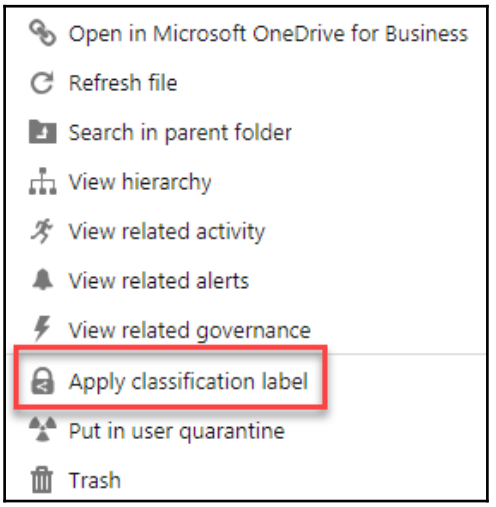
Cloud App Security file usage overview

7. Change the APP to **Salesforce**—your posted files are now viewable:



Data usage report of Salesforce

Some sensitive data was transferred unprotected to **Salesforce**, and from this point, we can define rules that this information is blocked if someone uploads it to such an application. If we change the filter to Microsoft OneDrive for Business and you upload the documents to these locations, **Cloud App Security** is already able to set the correct classification label manually or automatically on the information:



Applying classification labels with Cloud App Security

Now that we've seen a few powerful features of Cloud App Security, let's check out the last scenario: using Office 365 for data-leakage prevention.

Scenario 4 – Data leakage prevention in Office 365

In this section, we'll use an Exchange Online Mail flow rule to detect sensitive information. We'll encrypt the mail to protect the content automatically. This is a typical use case.

We create the Exchange Online Mail flow rules:

- **Rule 1**: Encrypts any email that contains a credit card number if the mail leaves the organization:

```
# Open an evaluated PowerShell
# Provide your global administrator credentials
$Creds = Get-Credential
$Session = New-PSSession -ConfigurationName
Microsoft.Exchange -ConnectionUri
https://outlook.office365.com/powershell-liveid/ -
Credential $Creds -Authentication Basic -AllowRedirection
Import-PSSession $Session
New-TransportRule -Name "Protect external mails (Contains
Credit Card)" -SentToScope NotInOrganization -
ApplyRightsProtectionTemplate "Encrypt" -
MessageContainsDataClassifications @(@{Name="Credit Card
Number"; minCount="1"})
```

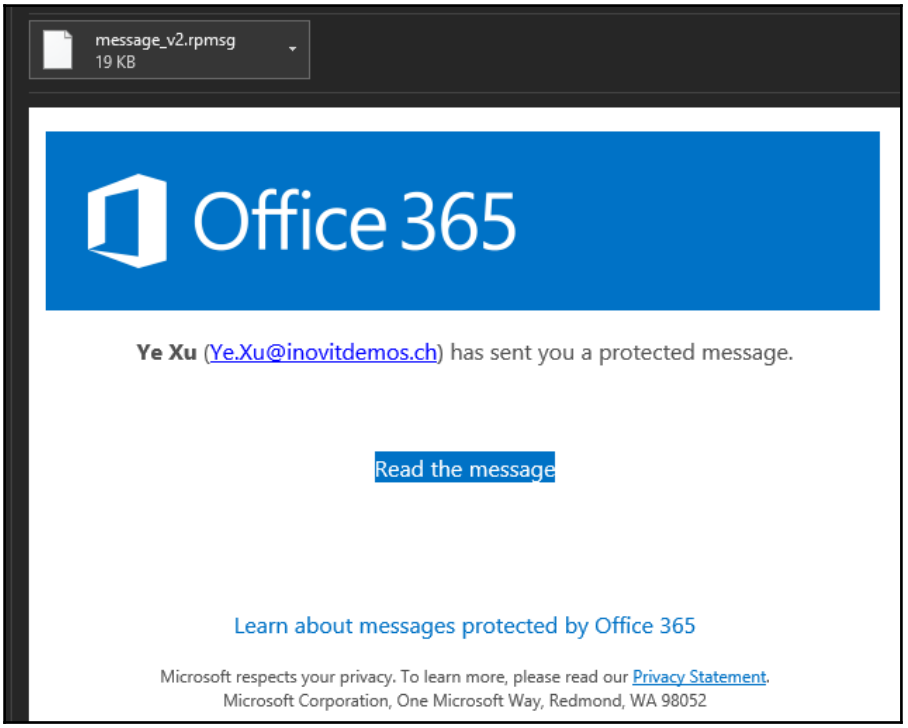
- **Rule 2**: Prevents emails from being sent externally if they're classified as `Confidential / All Employees`:

```
# Label ID of Confidential / All Employees gathered on the
Azure Information Protection blade
$labelid = "6eae6a7b-f321-4fc4-8049-1ef7cc9575b2"
$label = "MSIP_Label_"+$labelid+"_enabled=true"
New-TransportRule -name "Protect External User Access" -
SentToScope notinorganization -HeaderContainsMessageHeader
"msip_labels" -HeaderContainsWord $label -
RejectMessageReasonText "Internal Protected Message"
```

1. Log on to one of the test clients with `ye.xu@yourdomain1.com`. Ye Xu works in Sales.
2. Send a mail with the following content to `Karim.Manar@yourdomain1.com` and your private email address. Karim is a Controller.

   Hi Karim
   My AMEX card number is 344047014854133. The expiration date is 09/28, and the CVV is 4368
   Best regards
   Ye

3. Log in as `Karim.Manar@yourdomain1.com` on the second client and view the email.
4. Log in to your private account.
5. Click **Read the message** and you can read the message:



OMEv2 user experience

6. Log in to one of the test clients with `Dan.Jump@yourdomain1.com`. Dan is the CEO.
7. Classify an email with the following content as **Confidential | All Employees**, then send it to your private email account and to Ye Xu:

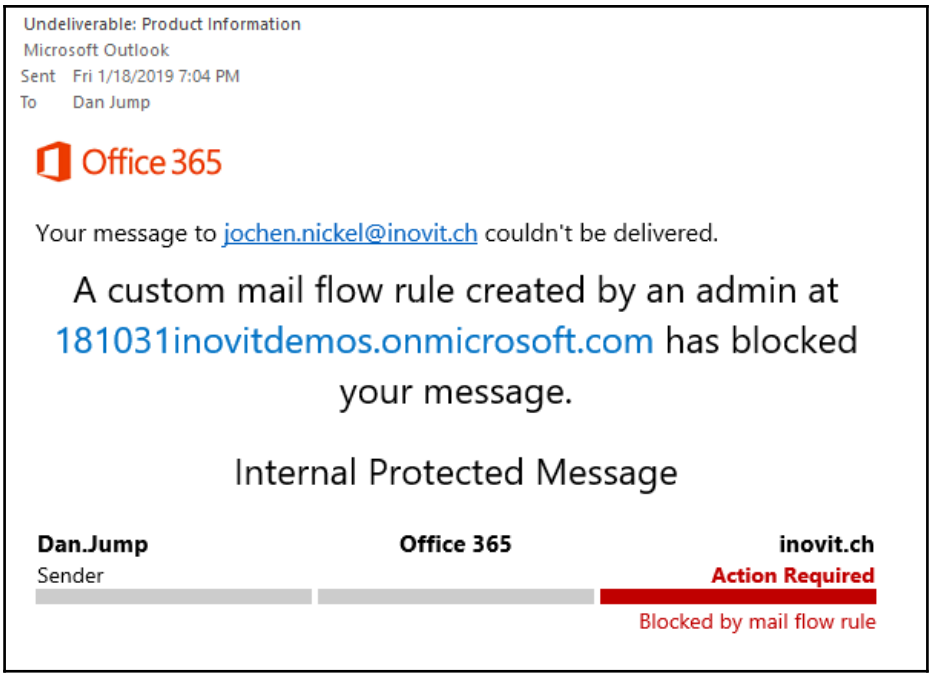
Hi Ye,

We're planning a big merger to create a new product. The code name for the product is "Identity Director".

It will be a revolution and a new market-leading product in the field of Identity and Access Management in the cloud.

Regards,
Dan Jump
CEO

8. You should receive a message similar to this:

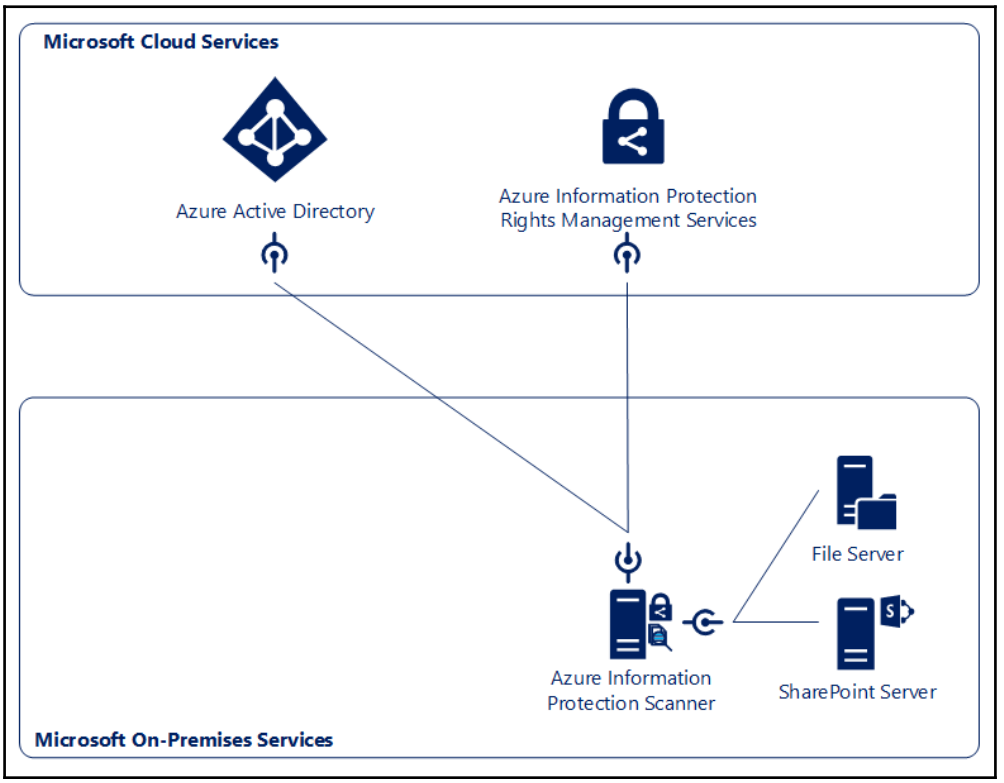


Office 365 DLP message

With this scenario, we saw a few capabilities of the Office 365 service, which helps us to discover and identify sensitive content. In the next section, we'll explore the functionalities of the Azure Information features for data at rest.

Understanding and using AIP capabilities for data at rest

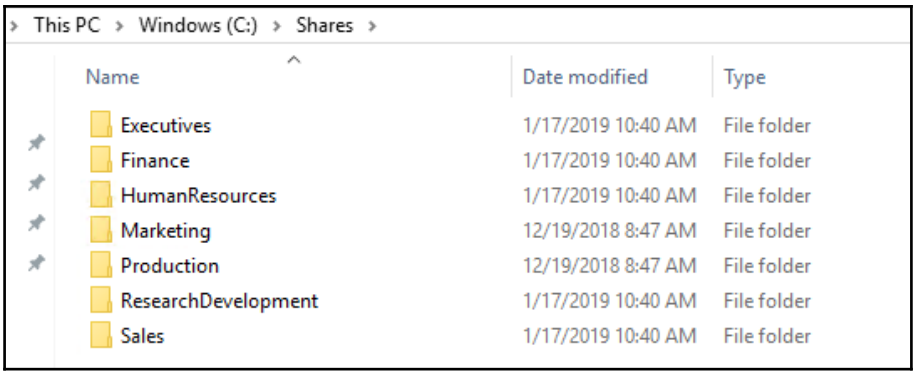
The identification and detection of sensitive information that isn't actively moved is a very important component inside an information-protection solution. For this reason, Microsoft provides the Azure Information Protection scanner, which allows you to scan two typical file locations, File Shares and SharePoint Document libraries, as you can see in the following diagram:



Azure Information Protection scanner architecture and components

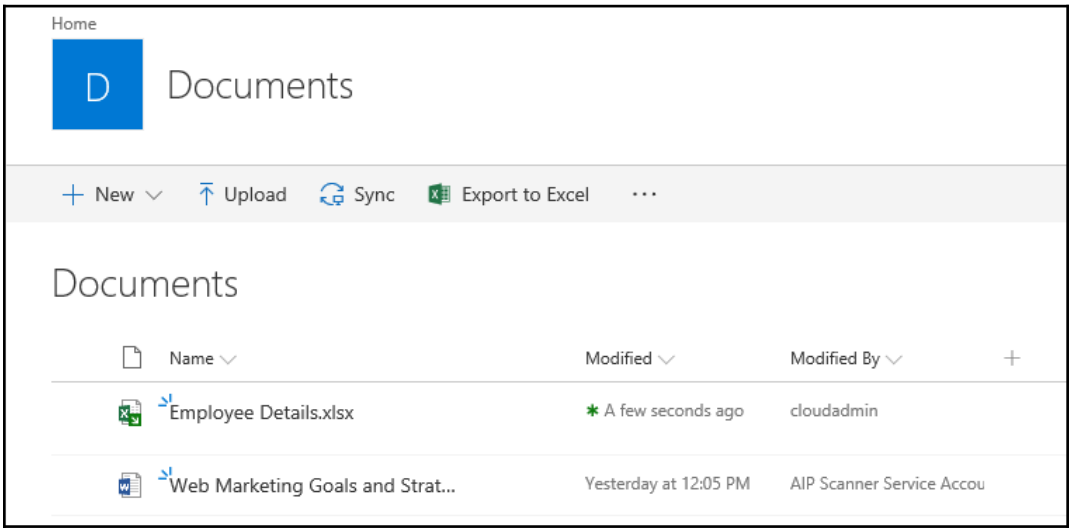
To explore the functionality of the **AIP Scanner**, we need to distribute some of our example documents to the following file locations:

1. Move the example contents from the code package to the **YD1INF01** server where we created the following test shares:



Example file structure

2. Upload some of the test files to your document library on SharePoint:



Sample data on SharePoint

We need to install and configure the **AIP Scanner** on our **YD1APP01** server where, in `Chapter 7`, *Deploying Solutions on Azure AD and ADFS*, we installed the SQL server:

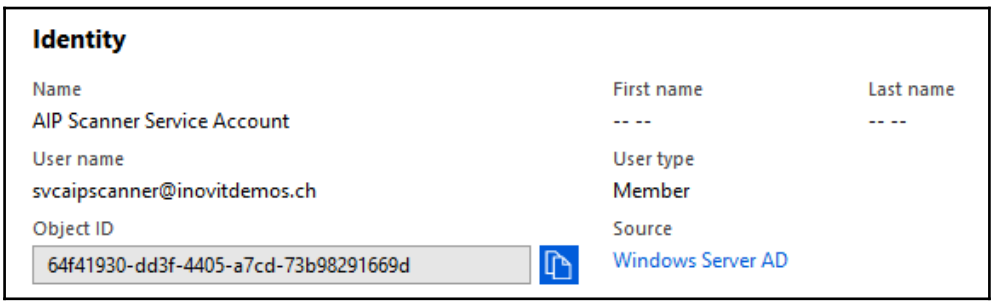
1. Use the following command to create the service account with which the AIP Scanner service runs and scans:

```
New-ADUser -Name "svcaipscanner" -SamAccountName
svcaipscanner -UserPrincipalName
svcaipscanner@inovitdemos.ch -path
"OU=Users,OU=AIP,OU=Managed Service
Objects,DC=inovitdemos,DC=ch" -AccountPassword (ConvertTo-
SecureString "YourPassword" -AsPlainText -Force) -Enabled
$True
```

   If you want to work with a cloud-only account, use the following command:

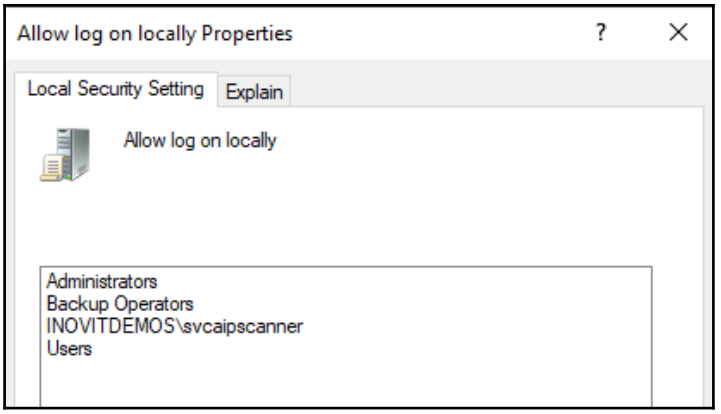
```
Connect-AzureAD
$PasswordProfile = New-Object -TypeName
Microsoft.Open.AzureAD.Model.PasswordProfile
$PasswordProfile.ForceChangePasswordNextLogin = $false
$Password = Read-Host -assecurestring "Please enter
password for cloud service account" $Password =
[System.Runtime.InteropServices.Marshal]::PtrToStringAuto(
[System.Runtime.InteropServices.Marshal]::SecureStringToBS
TR($Password)) $PasswordProfile.Password = $Password
$Tenant = Read-Host "Please enter tenant name for
UserPrincipalName (e.g. inovitdemos.ch)" New-AzureADUser -
AccountEnabled $True -DisplayName "AIP Scanner Cloud
Service" -PasswordProfile $PasswordProfile -MailNickName
"AIPScannerCloud" -UserPrincipalName
"AIPScannerCloud@$Tenant"
```

2. Synchronize the service account to your Azure AD:



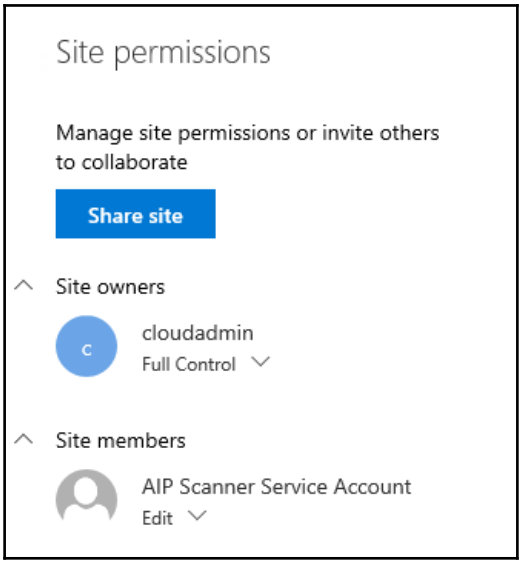
Synchronized AIP Scanner service account in Azure AD

3. The service account needs the following rights on different services:
    - Log on locally (need to be assigned) and Log on as service right (accomplished by the installation):



Log on locally permission assignment

    - Read (Discovery) or Contribute rights on the document library for classification/protection:



SharePoint access rights for the AIP Scanner service account

The following list shows the needed permissions:

- Read permissions to each file share repository for discovery, and Read/Write permissions for classification/protection
- Local administrator of the server and permissions to write to the SQL Server master database, **SQL specific permissions** if you can't grant the Sysadmin rights for the installation
- AzInfoProtectionScanner database needs to be created manually

The following accounts need to be `db_owner`:

- Service account for the scanner
- User account for scanner installation
- User account for scanner configuration
- For labels that re-protect or remove protection, the account needs to be part of the super users group

4. Install the **Azure RMS PowerShell** module with the following command:

```
Install-Module AADRM
```

5. Connect to Azure RMS with the following cmdlet and global administrator credentials:

```
Connect-AadrmService
```

6. Enable the **Azure RMS Super User** feature, which is disabled by default:

```
Enable-AadrmSuperUserFeature
```

7. Create a **mail-enabled group** called in your Azure AD and assign the group as `SuperUserGroup`:

```
Set-AadrmSuperUserGroup -GroupEmailAddress
"AzureRMSSuperUsers@yourdomain1.com"
```

We will discuss the **Super Users Group** in more depth in `Chapter 15`, *Configuring Azure Information Protection Solutions*.

8. Download the **AzInfoProtection.exe** binary from `https://bit.ly/2ccqSu0`.
9. Run the binary installation on the **YD1APP01** server.
10. Install the AIP Scanner on the server:

```
$cred = Get-Credential
Install-AIPScanner -SqlServerInstance YD1APP01 -
ServiceUserCredentials $cred
```

11. To run the configuration tasks, install the `AzureADPreview` PowerShell Module:

```
Install-Module AzureADPreview
```

12. Run the following command to **connect to your Azure AD** and provide global administrator credentials:

```
Connect-AzureAD
```

13. Create the `WebApp` and the associated `Service Principle`:

```
New-AzureADApplication -DisplayName AIPOnBehalfOf -ReplyUrls
'http://localhost'
$WebApp = Get-AzureADApplication -Filter "DisplayName eq
'AIPOnBehalfOf'"
New-AzureADServicePrincipal -AppId $WebApp.AppId
$WebAppKey = New-Guid
$Date = Get-Date
New-AzureADApplicationPasswordCredential -ObjectId
$WebApp.ObjectID -startDate $Date -endDate $Date.AddYears(1) -
Value $WebAppKey.Guid -CustomKeyIdentifier "AIPClient"
```

14. Build the `RequiredResourceAccess` object that's needed to **automate delegation** of permissions for the native application:

```
$AIPServicePrincipal = Get-AzureADServicePrincipal -All $true
| ? {$_.DisplayName -eq 'AIPOnBehalfOf'}
 $AIPPermissions = $AIPServicePrincipal | select -expand
Oauth2Permissions
 $Scope = New-Object -TypeName
"Microsoft.Open.AzureAD.Model.ResourceAccess" -ArgumentList
$AIPPermissions.Id,"Scope"
 $Access = New-Object -TypeName
"Microsoft.Open.AzureAD.Model.RequiredResourceAccess"
 $Access.ResourceAppId = $WebApp.AppId
 $Access.ResourceAccess = $Scope
```

15. Run the following command to create the `Native App` and associated Service Principle:

```
New-AzureADApplication -DisplayName AIPClient -ReplyURLs
http://localhost -RequiredResourceAccess $Access -PublicClient
$true
 $NativeApp = Get-AzureADApplication -Filter "DisplayName eq
'AIPClient'"
 New-AzureADServicePrincipal -AppId $NativeApp.AppId
```

16. Build the `Set-AIPAuthentication` command, to run it under the AIP Scanner service account:

```
"Set-AIPAuthentication -WebAppID " + $WebApp.AppId + " -
WebAppKey " + $WebAppKey.Guid + " -NativeAppID " +
$NativeApp.AppId | Out-File ~\Desktop\Set-
AIPAuthentication.txt
 Start ~\Desktop\Set-AIPAuthentication.txt
```

17. Open a PowerShell (Run as different user) and provide the on-premises AIP Scanner Service account credentials.
18. Run the command from the `Set-AIPAuthentication.txt` file, as follows:

```
Set-AIPAuthentication -WebAppID 10fea33d-a6c0-44cb-88ea-
eca3cf673d4d -WebAppKey f84ec310-cb36-44f9-ab7f-4edeecf099d0 -
NativeAppID 4f478966-0930-4fc3-b1e4-a3acb92d4932
```

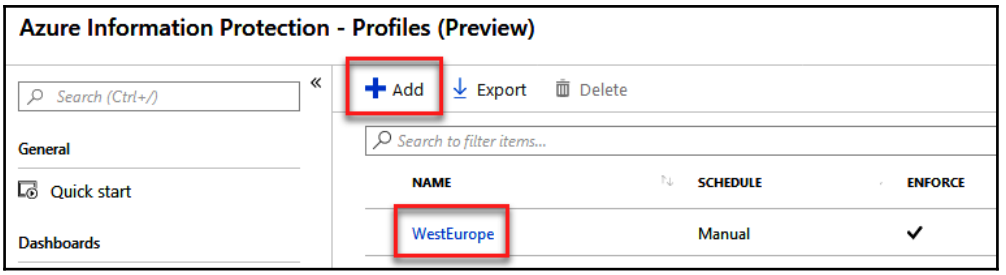
19. Accept to acquire an **authentication token**.
20. Restart the AIP Scanner.

In the next procedure, we will provide the steps to **upgrade** the AIP Scanner from a GA version to the newest preview version, `AzInfoProtection_PREVIEW_1.45.32.0.exe`:

1. Download the newest preview from `https://bit.ly/2ccqSu0` and install the binaries.
2. Update the scanner with the `Profile` parameter:

```
Update-AIPScanner -Profile WestEurope
```

3. To check that the scanner successfully restarts, **create the profile** in the Azure Information Protection blade:

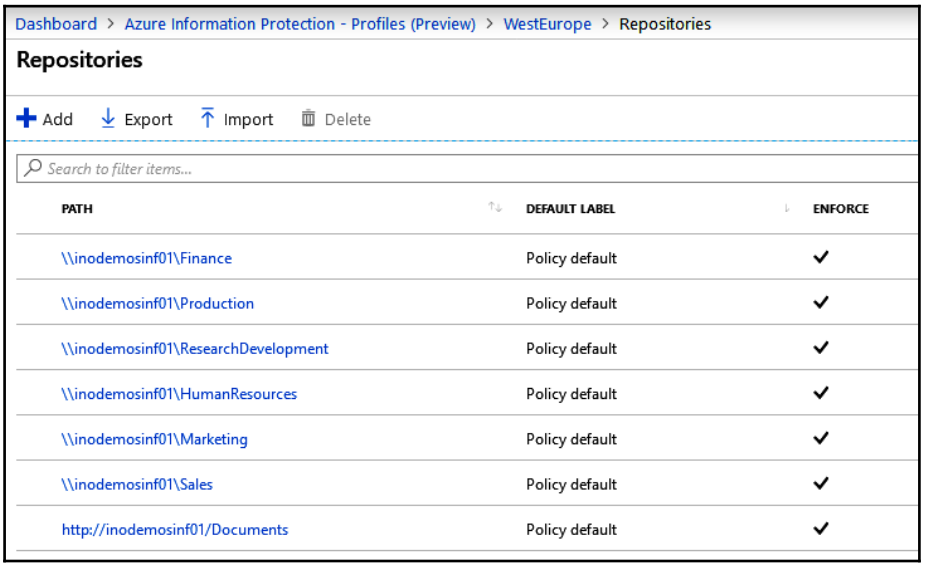


AIP Profiles for location handling

4. The database will also be renamed to the profile name:

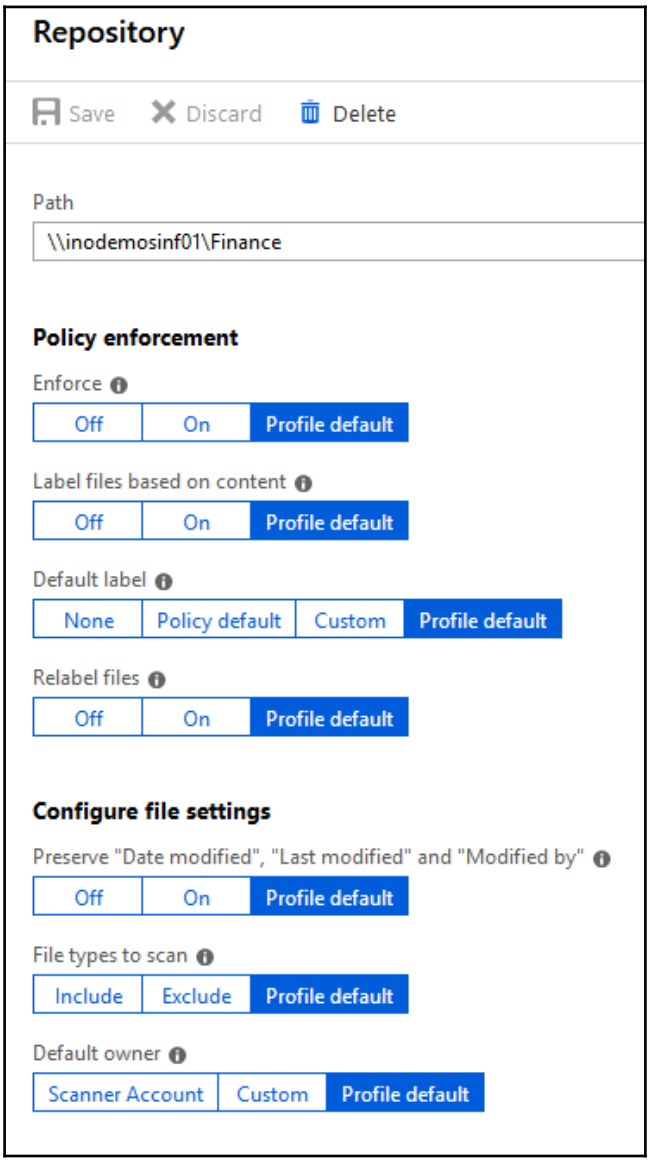
AIP Scanner database viewed in SQL Management Studio

5. Open the newly-created profile and configure the AIP Scanner settings
6. Configure the following repositories (replace the server name):



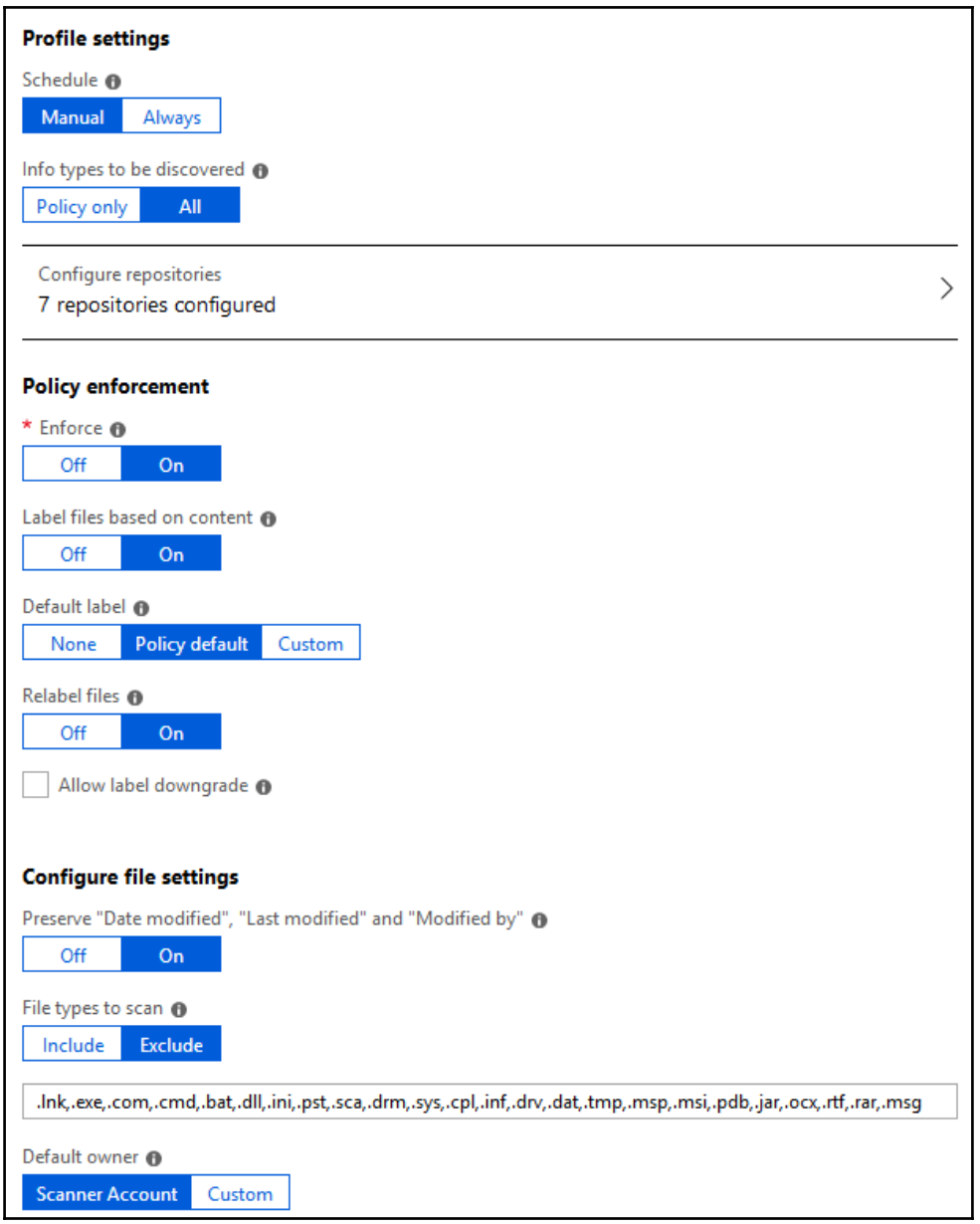
Configured AIP Scanner repositories

7. Use the following configuration for all the file-share repositories and the SharePoint document library:



Repository policy settings options

8. Use the following settings for the **Profile**:

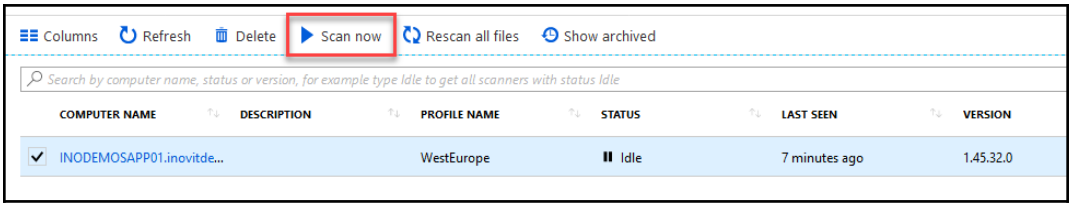


Profile settings section for the AIP Scanner

9. Navigate to the **Policies** settings under the **Classifications** section.
10. Click the **Global policy** and change the default label to **General**:

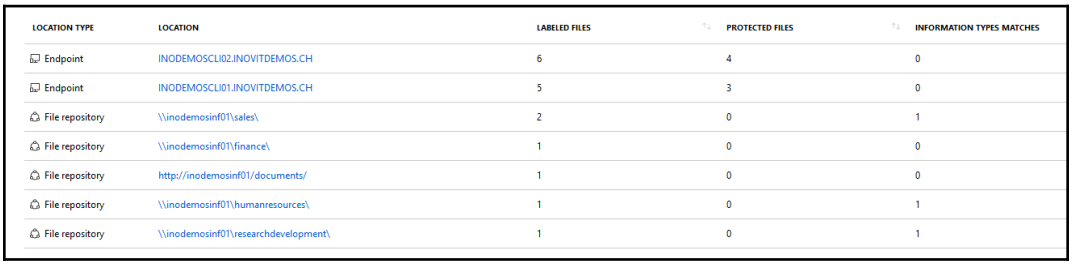
Default label configuration in AIP Global Policy

11. Navigate to the **Scanner** | **Nodes** section, mark the scanner server, and click **Scan now**:



Enabling a scan on the repositories

12. Navigate to **Dashboards** | **Data discovery (Preview)** to view the first results:



LOCATION TYPE	LOCATION	LABELED FILES	PROTECTED FILES	INFORMATION TYPES MATCHES
Endpoint	INODEMOSCLI02.INOVITDEMOS.CH	6	4	0
Endpoint	INODEMOSCLI01.INOVITDEMOS.CH	5	3	0
File repository	\\inodemosinf01\sales\	2	0	1
File repository	\\inodemosinf01\finance\	1	0	0
File repository	http://inodemosinf01/documents/	1	0	0
File repository	\\inodemosinf01\humanresources\	1	0	1
File repository	\\inodemosinf01\researchdevelopment\	1	0	1

Discovery results from the scanner

Read the following articles for information on customizing your configuration to your needs: `https://bit.ly/2T9CPn6` and `https://bit.ly/2Dk2YdK`.

You can also use all the discussed technologies to gather information about data at rest. But, it's very important to understand that everything depends on the quality of your detection rules that you can get from the data in motion and the associated processes. We will follow up on this rule in `Chapter 15`, *Configuring Azure Information Protection Solutions*.

Summary

In this chapter, we discussed and configured the key technologies for the discovery of sensitive data in motion, transit, and at rest. You saw how important these tasks are, and the benefits and data control they provide in your environment. They give you the power to build efficient detection and data-leakage-prevention rules. We'll use this knowledge to build an Azure Information Protection solution in `Chapter 15`, *Configuring Azure Information Protection Solutions*. But first, we'll learn more about the Azure RMS keys, which you need to understand to follow the protection process and find the right tool to troubleshoot against errors.

In the next chapter, you'll learn how Azure RMS keys are used and which deployment models are available for different compliance requirements.

14
Understanding Encryption Key Management Strategies

In this chapter, you'll learn to use the three different key deployment models to address different compliance requirements and understand what role the Azure Key Vault service plays in this. We'll discuss the three Azure Rights Management Services flows for a better understanding of how keys are used in the complete **Azure Information Protection** (**AIP**) solution to address the correct implementation and to help you troubleshooting the solution. This chapter will be divided into the following sections:

- Azure Information Protection key basics:
    - Key deployment models
    - What is a **hardware security module** (**HSM**)?
    - What is the Azure Key Vault?
- How Azure RMS works under the hood:
    - Algorithms and key lengths
    - User environment-initialization flow
    - Content protection flow
    - Content consumption flow

Let's start with the AIP key basics!

Azure Information Protection key basics

The Azure **Rights Management Service** (**RMS**) is part of the AIP solution of Microsoft. The Rights Management web service provides protection functionality, including administration, account certification, and licensing. Certification refers to the account certification and activation activities performed by Azure RMS. Each user must acquire a set of certificates and associated keys to be able to participate. Licensing refers to the set of operations by which Azure RMS grants access to protected content to authorized users. The Azure RMS service grants a use license (usage rights) for each document to authorized users.

The Azure RMS service exposes **Simple Object Access Protocol** (**SOAP**) interfaces that are used by clients to interact with Azure RMS

The Azure Rights Management web service also uses rights management templates, which specify a predefined set of rights and conditions that can be applied to protect content. In AIP, the rights-management templates are associated with labels. To support other classification solutions, rights-management templates can also be configured without the association of a label. We already dived into these topics in `Chapter 13`, *Identifying and Detecting Sensitive Data*.

The web service provides by default the needed high availability with a **Service Level Agreement** (**SLA**) of 99.9% uptime.

Azure RMS is an Azure service that provides information protection by using encryption to help secure documents, files, emails, and other content.

By default, AIP and the rights-management part generate the tenant key and manage most aspects of the tenant key life cycle. This is the simplest option with the lowest administrative overhead. In most cases, organizations with no additional or special security requirements will use this deployment model. They just sign up for AIP and the key-management process is handled by Microsoft.

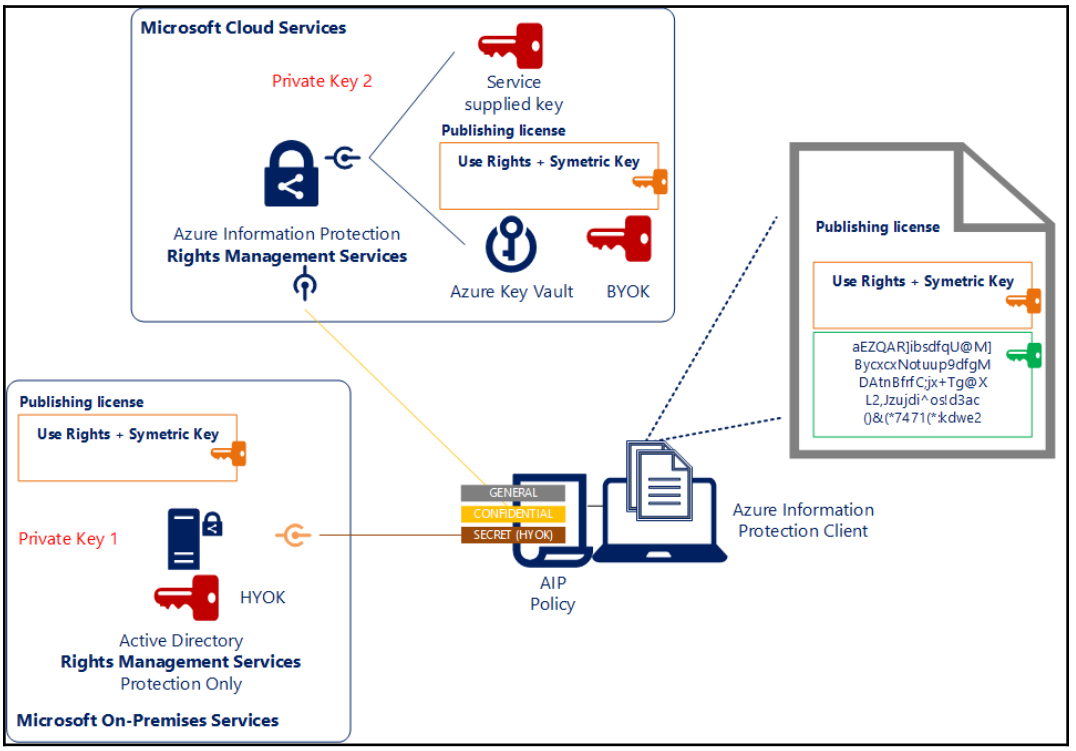
Alternatively, organizations might want to have complete control over their tenant key due to security or compliance requirements. For this reason, Microsoft provides a separate deployment model called **bring your own key** (**BYOK**). Typical customers are based in the insurance or health sector. In this deployment model, customers can create hardware- or software-based keys. Hardware-based keys will be created on an HSM, which is the recommended option, and will be securely transferred to and stored in Azure Key Vault. AIP will be configured to use customer keys. Many large organizations already have the hardware, software, and processes in place to manage their key material—this is typical on HSM. Most of them prefer to extend the HSM infrastructure to the cloud.

The third deployment model is called **hold your own key** (**HYOK**). This model requires an isolated on-premise **Active Directory Rights Management Service** (**AD RMS**) instance to be installed on a Windows Server. The AD RMS instance will work with a different private key to protect secret data. Protection using this AD RMS instance is based on labels. AD RMS, by default, doesn't provide any labeling capabilities.

Choosing between the different models is driven by your organization's compliance requirements, which are as follows:

- Contracts with your customers with a specific requirement for cryptography and audit
- Industry, corporate, and regulation requirements
- Data-sovereignty laws, which bring place regulations and key placement
- Data classification, which defines what sensitive data can't be stored in the cloud

The following diagram shows the three deployment models:



Azure RMS deployment models

Within the following licenses, you can use the three deployment models:

Scenario	Office 365 E3 or higher	AIP P1 or EMS E3/M365 E3	AIP P2 or EMS E5/M365 E3
Microsoft-managed	X	X	X
BYOK	X	X	X
HYOK			X

There is also an option to use service encryption with a customer key in Office 365 to provide and control the encryption keys for your at-rest Office 365 data at the application level.

The following tenant key actions are available:

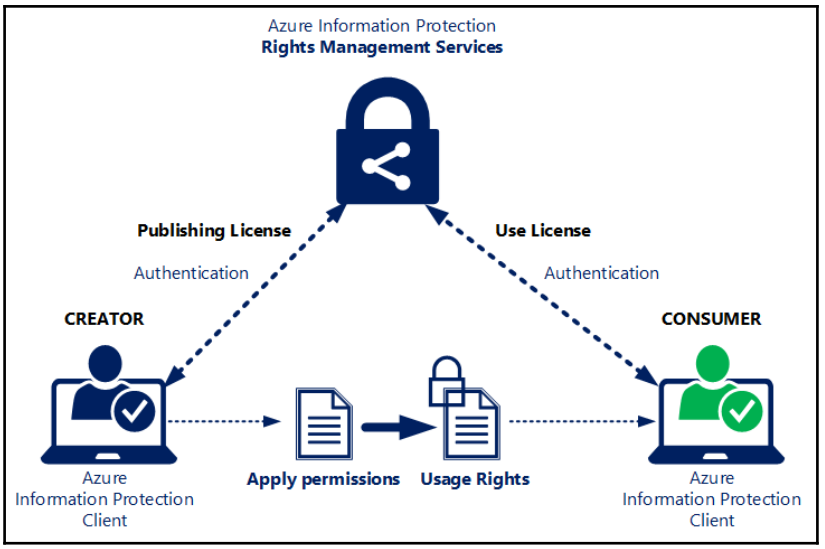
Action	Microsoft-managed	Customer-managed
Revoke		X
Rekey	X	X
Back up and recover		X
Export	X	
Breach response	X	X

To find out more about service encryption with Customer Key for Office 365, visit `https://bit.ly/2SxCJW5`.

Also shown in the preceding diagram are the three keys that matter in the main protection flow:

- **Red**: The tenant private key (asymmetric), which is used for the decryption of the publishing license
- **Orange**: The tenant public key (asymmetric), which is used for the encryption of the publishing license
- **Green**: The document-specific key (symmetric), which is used for encryption/decryption of the content

To understand this better, check out the main functionality of the **Rights Management Services** in the following diagram:



Azure RMS basic functionality

The process starts with protecting information and ends with consuming it with the applied permissions from the data owner or the classification level:

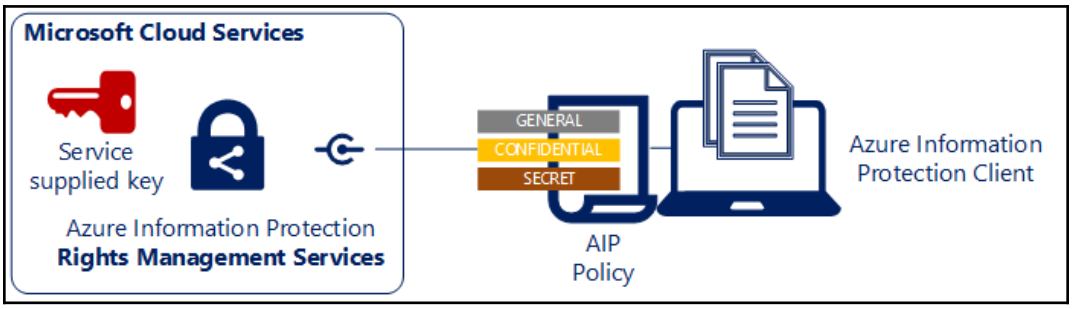
1. The document will be created, labeled, and protected with RMS—for this, we need to be authenticated against the rights-management service
2. The label-associated rights and the key material will be requested transparently from the user
3. The key material will be received transparently by the user
4. The document will be protected with the chosen rights
5. The document will be distributed to the required recipients
6. The document will be received from the recipient, so the recipient needs to be authenticated against the Rights Management Service by Single-Sign-On or Single Login
7. The key material and the usage rights will be requested transparent to the recipient
8. The key material will be received, and the recipient can access the information with the appropriated rights, otherwise the user will get an access denied message

Be sure to plan your key deployment carefully.

Next, we will discuss the Microsoft-managed key-deployment model.

Microsoft-managed keys

If AIP gets activated, a new tenant key be generated, stored, and managed by the Azure Information Protection services. In this default case, the key type is referred to as a Microsoft-managed key. The following diagram shows the deployment model:



Microsoft Services key model

You should use this deployment model if Microsoft's security and controls are adequate for your organization. This model is available by default and doesn't need any additional subscription or configuration. There are also no special plans needed for capacity, performance, or scale. For most organizations, this is the best option. The deployment model has the following features:

- Completely-managed tenant private key
- Tenant private key is encrypted at rest and handled as customer data
- Default key length is 2,048 bit RSA

Now, we will verify the `Microsoft-managed` key that's used by our tenant with the following cmdlets:

```
# Installing the AADRM module to administer Azure RMS
Install-Module -Name AADRM
# Connecting to Azure RMS with global administrator rights
Connect-AadrmService
```

```
# Listing the Azure RMS keys
Get-AadrmKeys
```

The following screenshot shows the expected result of a freshly-activated tenant:

```
PS C:\WINDOWS\system32> Get-AadrmKeys


KeyIdentifier  : a4fd8a9b-e573-48fb-afda-814f8ed31aec
CreationTime   : 10/31/2018 2:51:23 PM
Status         : Active
KeyType        : Microsoft-managed
FriendlyName   : INOVITDEMOS by inovit GmbH
PublicKey      :
KeyVaultKeyUrl :
```

Actual RMS key information

Another important factor is where your Azure RMS service is provisioned for choosing an Azure Key Vault location in a BYOK-deployment for latency reasons. We can gather this information with the following cmdlet:

```
Get-AadrmConfiguration
```

In our case, the Azure RMS tenant is provisioned in Europe, which we can see in the following screenshot:

```
PS C:\WINDOWS\system32> Get-AadrmConfiguration


BPOSId                                  : 7709ca2b-3be8-4d92-89d7-dc1e274b4d0e
RightsManagementServiceId               : 94ce956b-a527-4b88-9266-3901f927b033
LicensingIntranetDistributionPointUrl   : https://94ce956b-a527-4b88-9266-3901f927b033.rms.eu.aadrm.com/_wmcs/licensing
LicensingExtranetDistributionPointUrl   : https://94ce956b-a527-4b88-9266-3901f927b033.rms.eu.aadrm.com/_wmcs/licensing
CertificationIntranetDistributionPointUrl : https://94ce956b-a527-4b88-9266-3901f927b033.rms.eu.aadrm.com/_wmcs/certification
CertificationExtranetDistributionPointUrl : https://94ce956b-a527-4b88-9266-3901f927b033.rms.eu.aadrm.com/_wmcs/certification
AdminConnectionUrl                      : https://admin.eu.aadrm.com/admin/admin.svc/Tenants/94ce956b-a527-4b88-9266-3901f927b033
AdminV2ConnectionUrl                    : https://admin.eu.aadrm.com/adminV2/admin.svc/Tenants/94ce956b-a527-4b88-9266-3901f927b033
OnPremiseDomainName                     :
Keys                                    : {a4fd8a9b-e573-48fb-afda-814f8ed31aec}
CurrentLicensorCertificateGuid          : a4fd8a9b-e573-48fb-afda-814f8ed31aec
Templates                               : {026be843-becf-425f-a776-27d4c1b8fd54, 68ab2d70-f65e-4b83-b3fb-8ed3e7f06ee7}
FunctionalState                         : Enabled
SuperUsersEnabled                       : Disabled
SuperUsers                              : {}
AdminRoleMembers                        : {}
KeyRolloverCount                        : 0
ProvisioningDate                        : 10/31/2018 2:51:23 PM
IPCv3ServiceFunctionalState             : Enabled
DevicePlatformState                     : {Windows -> True, WindowsStore -> True, WindowsPhone -> True, Mac -> True...}
FciEnabledForConnectorAuthorization     : True
DocumentTrackingFeatureState            : Enabled
```

Azure RMS actual configuration overview

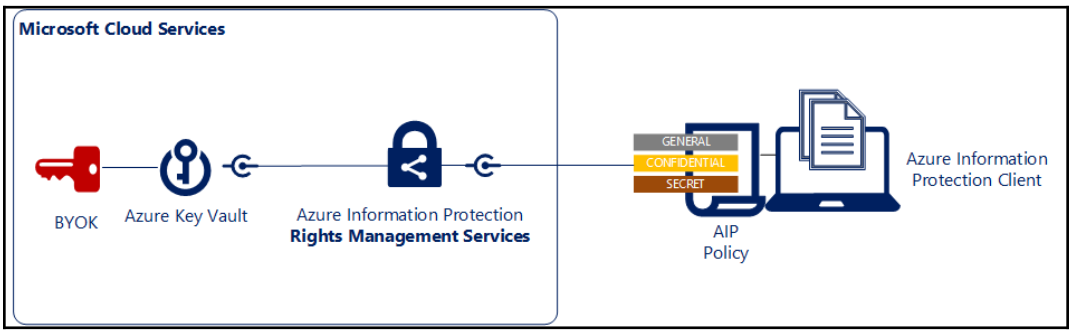
Next, we will take a deeper look into the BYOK deployment model.

Bring your own key

If you use Azure Key Vault to store key, which indicates that you use the BYOK deployment, you can import keys from your on-premises HSM or generate keys inside the **Azure Key Vault** that are stored in an HSM. Keys never leave the HSM boundaries.

At the time of writing, Microsoft provides a public preview using dedicated HSMs. You can find more information at `https://bit.ly/2KFnJT7`.

You should use the **BYOK** deployment model if you need to fulfill additional compliance requirements, such as data residency or specific cryptographic regulations. The following diagram shows the architecture of the deployment:



Bring your own key model

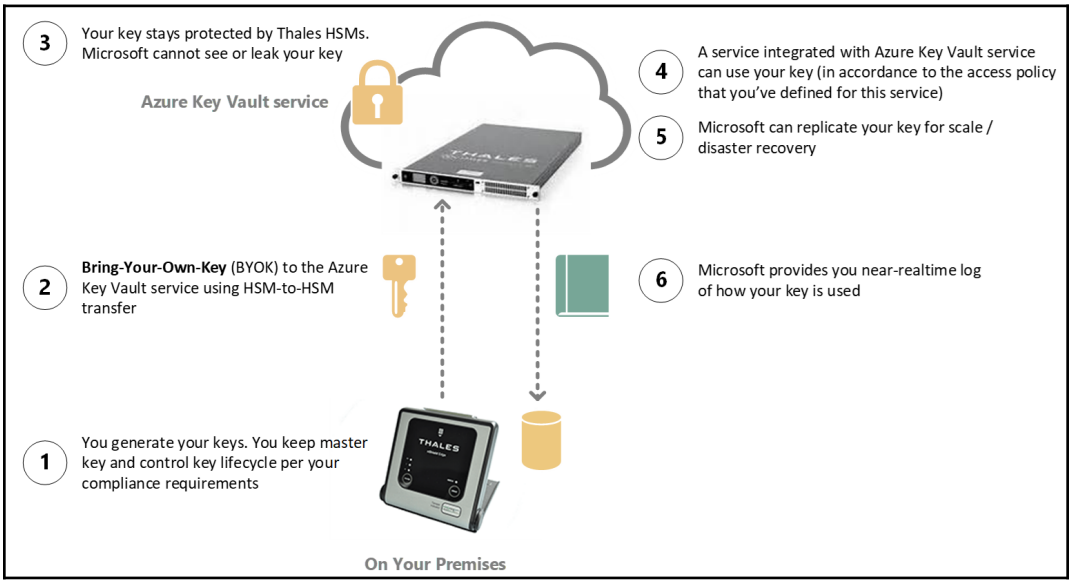
The following key facts are related to BYOK:

- Azure Information Protection doesn't have visibility over the private key; there is a clear separation of duties with the usage of **Azure Key Vault**
- Cryptographic operations are done completely within Azure Key Vault
- The tenant private key is stored in Azure Key Vault

With the configuration part, we'll take a closer look at HSMs and the Azure Key Vault. We'll configure our Azure-Key-Vault-based key and change to the BYOK deployment.

What is an HSM?

HSMs are physical devices that provide a hardened, tamper-resistant environment to manage and securely store digital keys used in Azure RMS and other applications. HSMs provide organizations with the ability to securely manage their private keys on-premises. The following diagram, provided by Thales, shows the BYOK-integration scenario:



Thales HSM integration

The following table provides a summary of the benefits of implementing a **Thales** HSM in the Azure RMS deployment:

Secure key storage	HSMs provide a tamper-resistant environment for the storage of private keys. All Thales HSMs are certified to meet the highest security standards.
Compliance	HSMs are FIPS 140-2 Level 3 Standard, which is the most widely accepted benchmark for hardware security in both enterprise and government environments.
Extensibility	Using a Thales HSM for key storage allows the Azure RMS environment to be extensible to on-premises for future migrations.

You can also use Gemalto HSMs for this scenario. You can find an example at `https://bit.ly/2GW3t1d`.

What is the Azure Key Vault?

Azure Key Vault can be best described as a Key Management Service that is based on FIPS 140-2 validated HSMs, which also provides secrets and certificate-management services. Azure Key Vault is used for Azure RMS BYOK deployments. The service itself provides the following capabilities:

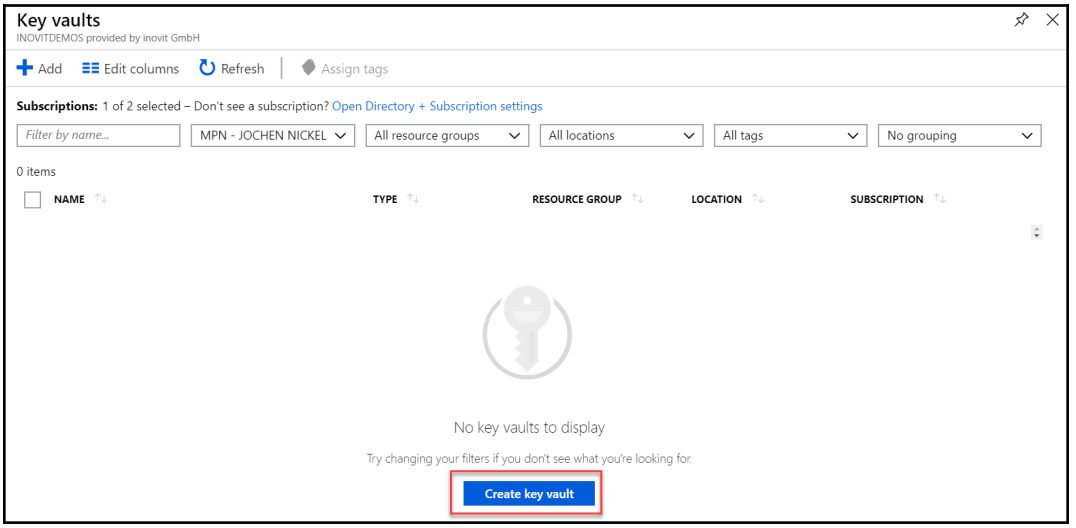
- Centralized secret management
- Compliance through Software/Hardware HSM protection
- Read/Write management over REST API/SDK/PowerShell and CLI
- Access and usage monitoring
- Automated distribution, logging inspection, and deployments
- Throttling and Versioning
- SLA 99.9% and 6 persistent copies (3 copies same region/3 additional copies secondary region)

For Azure RMS and BYOK, you need to use the P1 Premium option to import your keys from on-premises or create keys directly in Azure Key Vault. The term BYOK is used both times. For production environments, we recommend using an on-premises HSM, and you can work through the following source to configure your environment: `https://bit.ly/2wma4Yl`.

For demonstration purposes, we are using keys that were generated in Azure Key Vault, because we don't think you want to invest many thousands of dollars into your personal HSM.

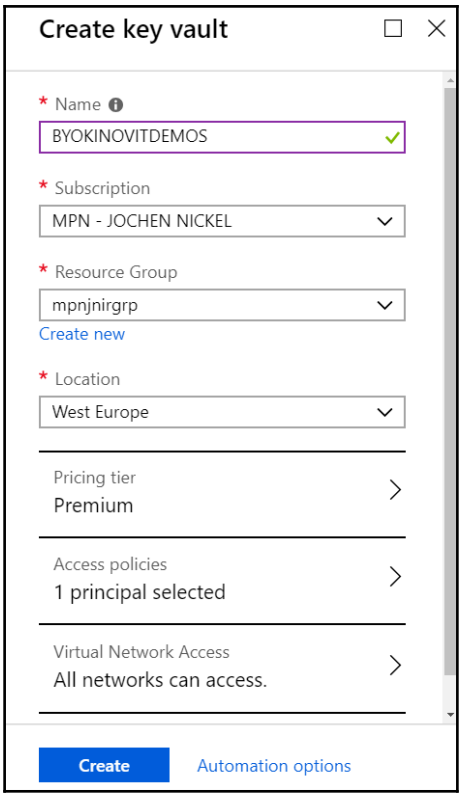
Perform the following steps to test the BYOK deployment in your demo environment:

1. Open `https://portal.azure.com` as a global administrator, choose **Key Vaults**, and click **Create key vault**:



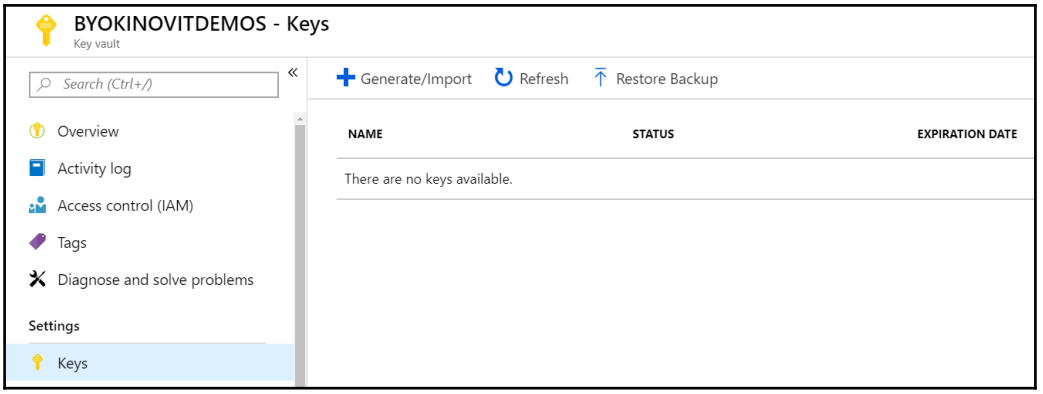
Key Vault creation

2. Fill in the following information:
   - Use the **Premium Pricing Tier**
   - Use the **Location** near your service to avoid network-latency issues
   - Leave the access policy and the **Virtual Network Access** defaults for now:



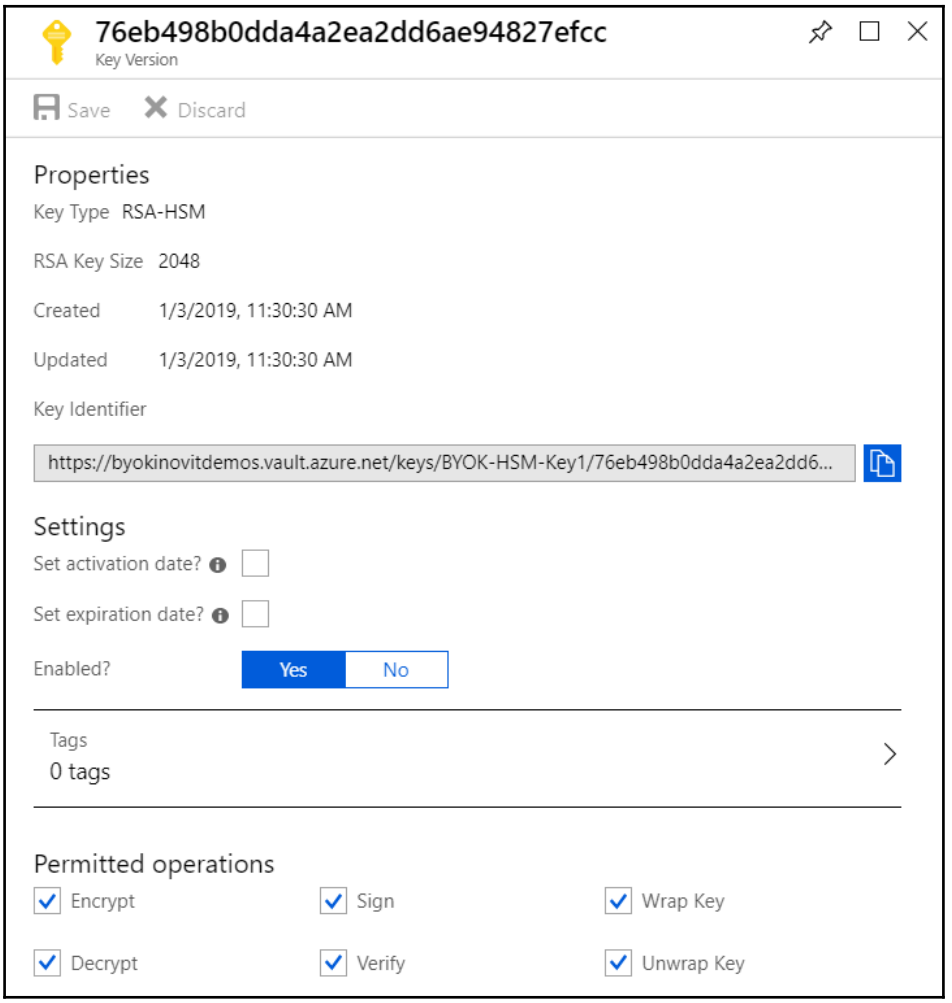
Key Vault properties

3. Generate the key using the **Generate/Import** option:



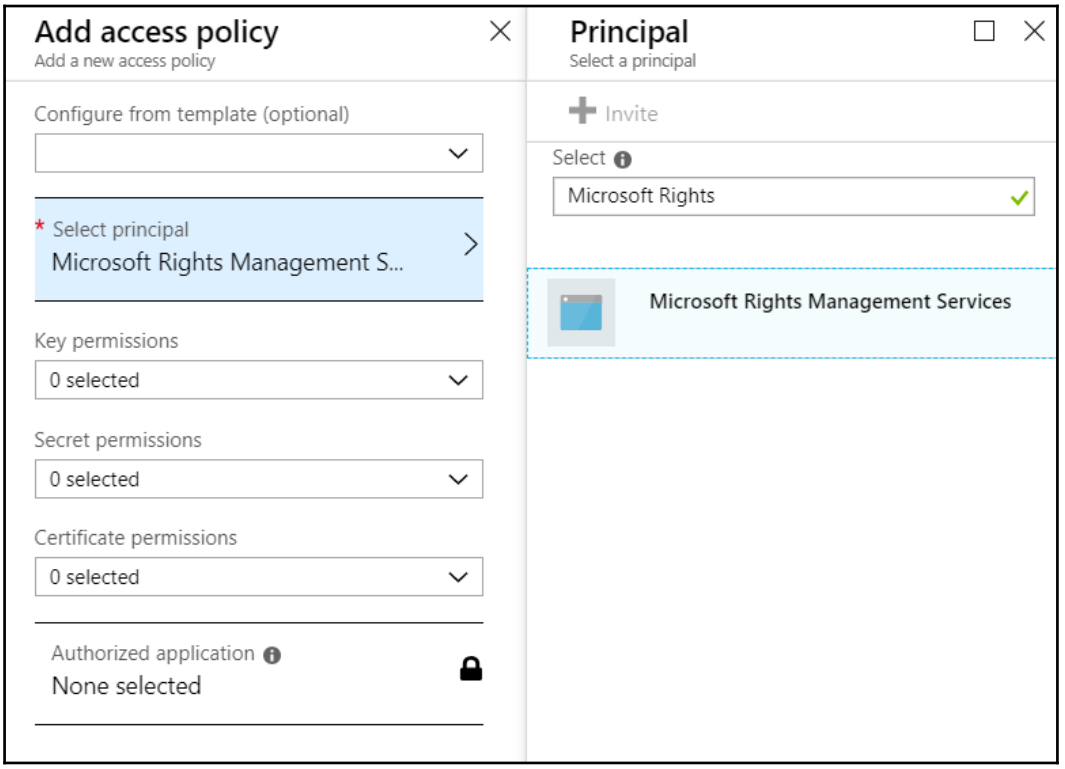
Key creation process

4. Use the following options:
    - **Key Type RSA-HSM** with a key size of **2048** bit
    - Copy the **Key Identifier** to a notepad for later:



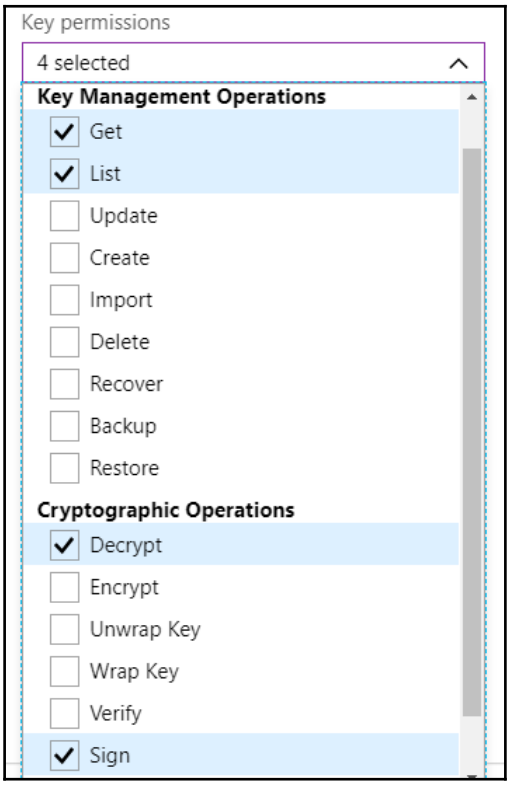
Key properties

5. Add the **Microsoft Rights Management Services** principal to your access policy:



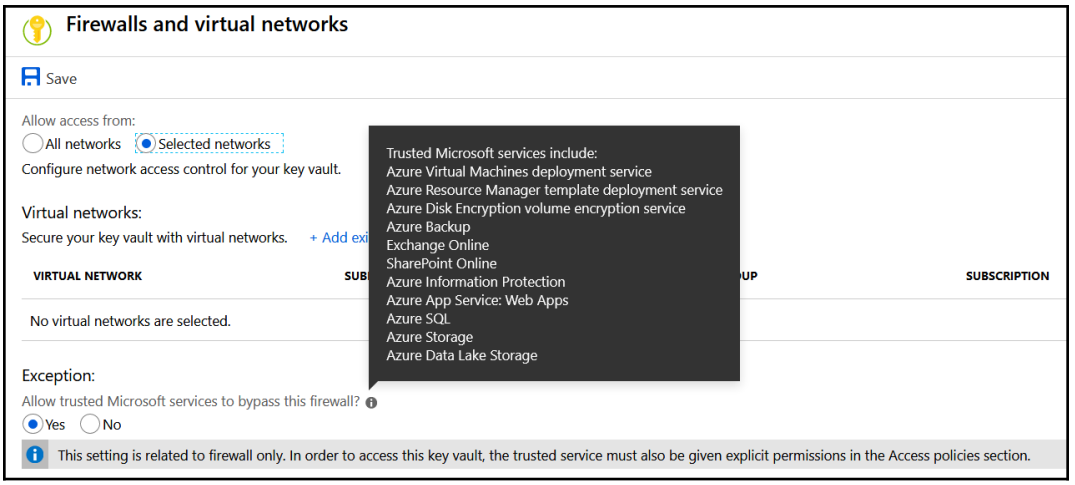
Service Principal assignment

6. Assign the following minimal permissions:
    - **Key Management Operations**: **Get** and **List**
    - **Cryptographic Operations**: **Decrypt** and **Sign**:



Permission assignment

7. Optionally, you can limit the access to your vault by using a new feature set—a firewall in front:



Key Vault firewall options

8. Configure the `Azure RMS` service to use our freshly-created key:

```
# Connecting to Azure RMS with global administrator rights
Connect-AadrmService

# Configure Azure RMS to use the key from our key vault
Use-AadrmKeyVaultKey -KeyVaultKeyUrl "<Your Key Identifier
from the notepad>" -FriendlyName FirstBYOKKey -Verbose
```

The output of the preceding command will look as follows:

```
PS C:\WINDOWS\system32> Use-AadrmKeyVaultKey -KeyVaultKeyUrl "https://byokinovitdemos.vault.azure.net/keys/BYOK-HSM-Key1/76eb498b0dda4a2ea2dd6ae94827efcc" -FriendlyName
tBYOKKey -Verbose
WARNING: This operation tells the Rights Management service to use the specified key in Key Vault.

Confirm
Are you sure you want to perform this action?
Performing the operation "Use-AadrmKeyVaultKey" on target "current organization".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): y
The Rights Management service successfully added the key.
```

Change key usage to the newly created one

Set the key to the `Active` state:

```
# View the new key with the "Archived state"
Get-AadrmKeys

# Next we set the key to the "Active" state
Set-AAdrmKeyProperties -KeyIdentifier "<Your Key
Identifier>" -Active $true
```

The output of the preceding command will look as follows:

```
PS C:\WINDOWS\system32> Set-AadrmKeyProperties -KeyIdentifier a9cfc5ea-00e4-4749-b768-625d421bf9a2 -Active $true
WARNING: This operation sets the selected key as the active key, and archives the currently active key.

Confirm
Are you sure you want to perform this action?
Performing the operation "Set-AadrmKeyProperties" on target "current organization".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): Y
The Rights Management service successfully added the key.
```

Activating the key

9. Your Azure Key Vault stored key is in use:

```
PS C:\WINDOWS\system32> Get-AadrmKeys


KeyIdentifier  : a4fd8a9b-e573-48fb-afda-814f8ed31aec
CreationTime   : 10/31/2018 2:51:23 PM
Status         : Archived
KeyType        : Microsoft-managed
FriendlyName   : INOVITDEMOS by inovit GmbH
PublicKey      :
KeyVaultKeyUrl :

KeyIdentifier  : a9cfc5ea-00e4-4749-b768-625d421bf9a2
CreationTime   : 1/3/2019 10:30:58 AM
Status         : Active
KeyType        : Customer-managed (BYOK)
FriendlyName   : FirstBYOKKey
PublicKey      : {"n":"tFFz0ml2e2GSzbQfvOttKXyeZ1ucNu4F5iJQuXd9B_3fN-mXtgt7S39MXSRC_x2lncgTVGXVjZ11Mv3g5yJ-Y7onanCdsogiV3ZoA-fPt1bT-wT8mVZ8F9rAgl6COHCxHlQWhfTX00sa7m47i0B2G
                 si2oYZ39W4QMaIPRv9Gx3hCM6pcrg0W0XEpSt_SVo-oEbw_8ooAWkBkqBFbNc7AvpfNQHcMMkqJ5342AS0UcUz2sfxl0pO5Aiq5MzRICPXZ9ZHb36ZZgChCKOXjGqySONUUTuk1VdGSpqITYehrPVz8zclO
                 IFFhAr1DI1rrn5yqNenE7z0iJEhix8uE2rUgww", "e":"AQAB"}
KeyVaultKeyUrl : https://byokinovitdemos.vault.azure.net/keys/BYOK-HSM-Key1/76eb498b0dda4a2ea2dd6ae94827efcc
```

Actual used keys overview

10. You can change back to the `Microsoft-managed key` with the following command:

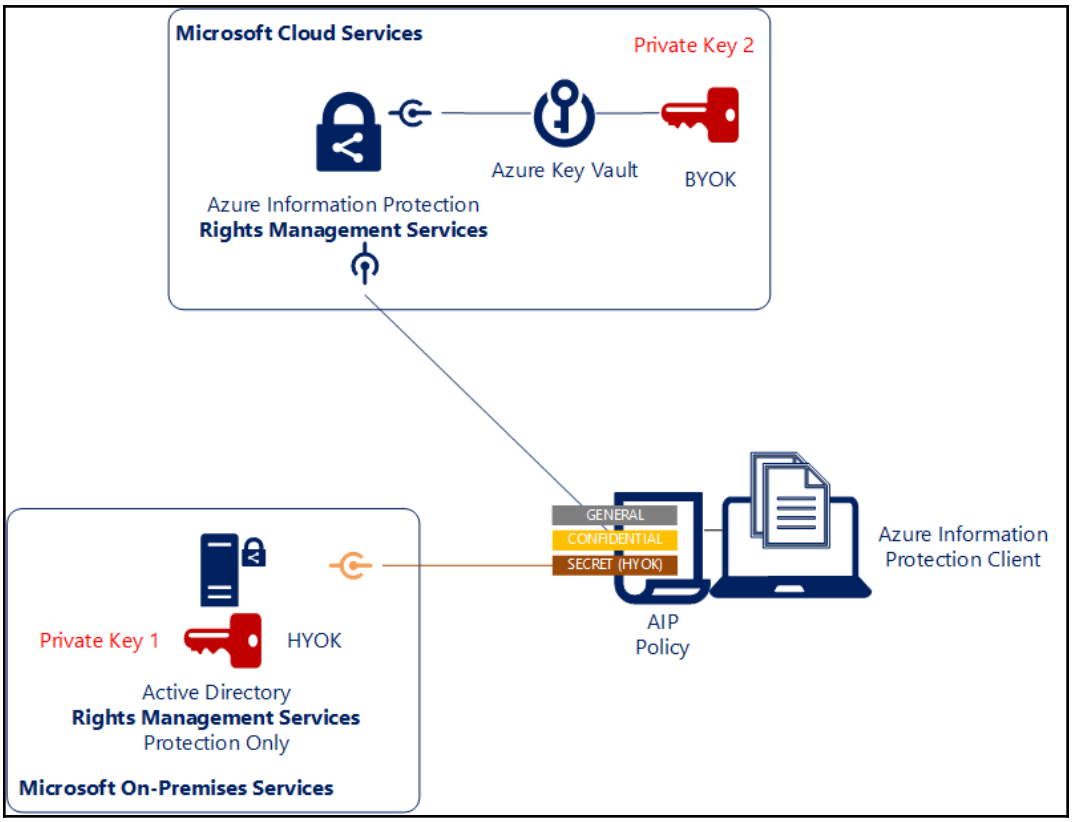
```
# Next we set the Microsoft-managed key to the "Active" state
Set-AAdrmKeyProperties -KeyIdentifier "<Your Microsoft-managed Key Identifier>" -Active $true
```

Now that we know about HSMs, Azure Key Vault, and how to configure a BYOK deployment, we'll jump into the HYOK deployment model in the next section.

Hold your own key

HYOK uses an isolated on-premises AD RMS instance that provides the RMS templates based on the second different private key that's driven by an AIP label. This deployment model should be chosen for high security and compliance requirements.

Most of the time, this us used for data that can't be stored on a public cloud. This sensitive data needs to be stored and protected on-premises. Keep in mind that HYOK-protected data is typically between 3 to 5% of an organization's protected data. The following diagram shows the deployment model:



Hold your own key model

The following limitations/benefits are available by design:

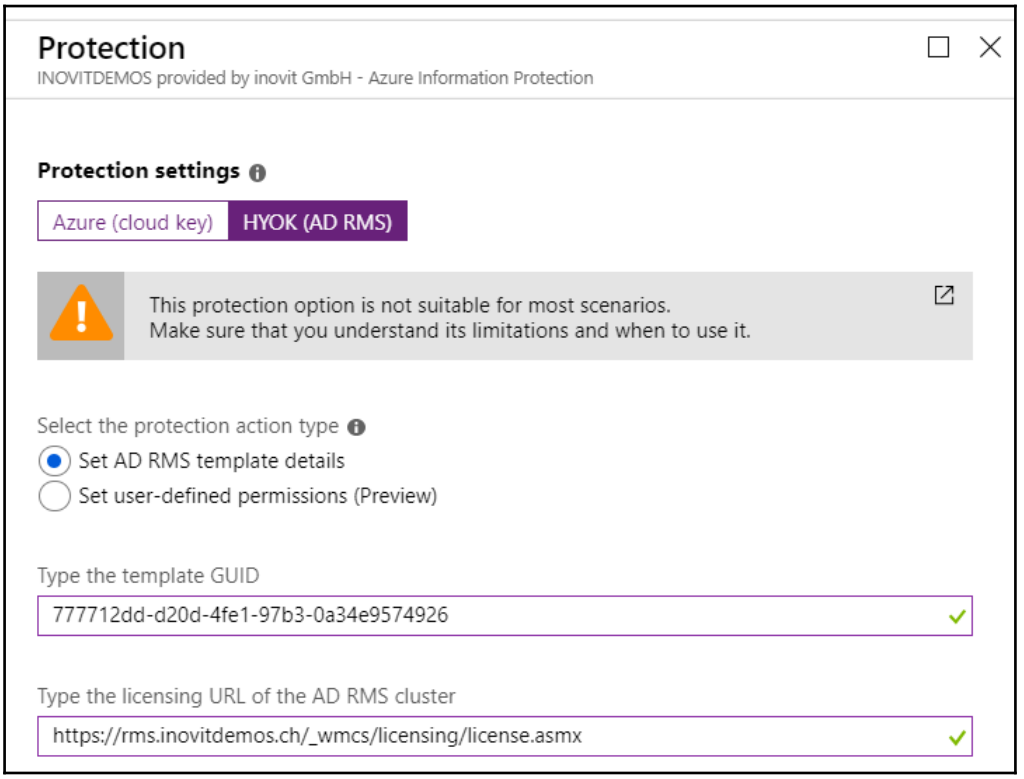
- External sharing configuration isn't available through the **Azure Information Protection** service
- External sharing is only possible in a controlled manner with known and named partners
- Office 365 services cannot provide indexing or search capabilities; Exchange Online transport rules and Office 365 DLP can't decrypt the content and inspect it

- Sharing and the access logs aren't disclosed to anyone
- Data remains encrypted always, inaccessible even to Microsoft services

To use the on-premises AD RMS infrastructure in your AIP labels, you need to choose the **HYOK (AD RMS)** option in the protection options. You can choose between two options:

- **Set AD RMS template details** (template GUID and licensing URL)
- **Set user-defined permissions (Preview)**

The following screenshot shows the template-based approach:



HYOK template configuration in a label

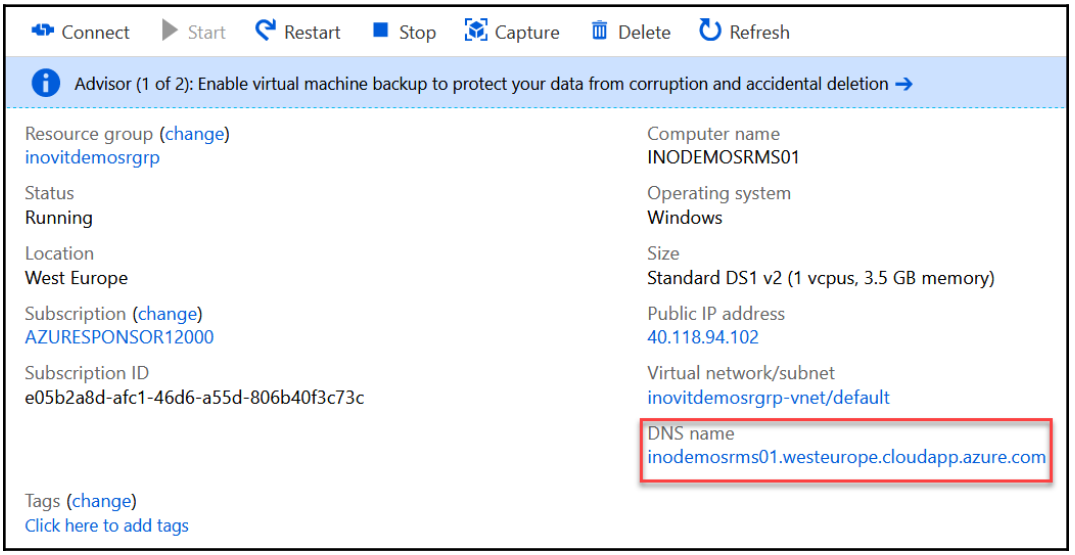
To configure the HYOK deployment in your demo environment, we'll need to add an additional virtual machine to your existing domain to install and configure AD RMS.

Keep in mind that the following configuration is just for proof-of-concept or demo purposes:

- For production environments, we recommend that you use a SQL Server for AD RMS
- Use the identity federation option to support external sharing if needed
- Use a dedicated administrative account
- Publish the service endpoints through a firewall or reverse proxy

With the following steps, we start the configuration of a local AD RMS infrastructure:

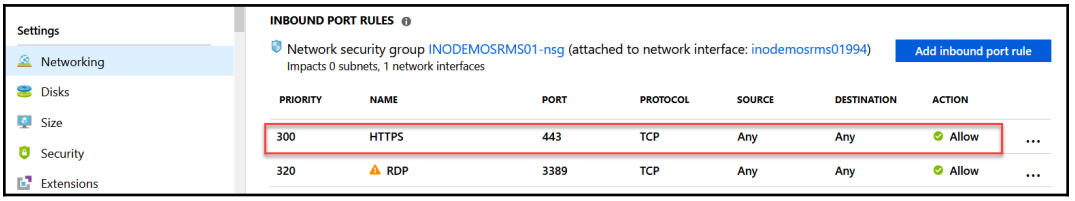
1. Create a Windows Server 2016/2019 Server virtual machine like the following:



VM creation to hold the AD RMS service

2. Create a CNAME entry in your public DNS configuration—in our example, this is it's `rms.inovitdemos.ch CNAME inodemosrms01.westeurope.cloudapp.azure.com`.

3. Open the virtual machine directly for **HTTPS** (PoC/Demo):



Firewall configuration to accept HTTPS traffic on the VM

4. You need a public SSL certificate at least with the RMS cluster URL included:
    - We use a wildcard certificate—`*.inovitdemos.ch`

Be sure that you join your virtual machine to your domain and install the certificate into the local machine store! For demo purposes, you can also run with an HTTP configuration.

5. Prepare the Active Directory objects as a domain administrator on the domain controller for the configuration.
6. Create the RMS cluster service account:

```
New-ADUser -Name "svcrmscluster" -SamAccountName
svcrmscluster -UserPrincipalName
svcrmscluster@inovitdemos.ch -path
"OU=Users,OU=AIP,OU=Managed Service
Objects,DC=inovitdemos,DC=ch" -AccountPassword (ConvertTo-
SecureString "MIA@me1976ch" -AsPlainText -Force) -
EmailAddress "svcrmscluster@inovitdemos.ch" -Enabled $True
```

7. Create the RMS superusers group and apply an email address to recover protected data:

```
New-ADGroup -Name "AIP RMS Super Users" -SamAccountName
"AIP RMS Super Users" -GroupCategory Security -GroupScope
Global -DisplayName "AIP RMS Super Users" -Path
"OU=Groups,OU=AIP,OU=Managed Service
Objects,DC=inovitdemos,DC=ch" -Description "AIP RMS Super
Users"
```

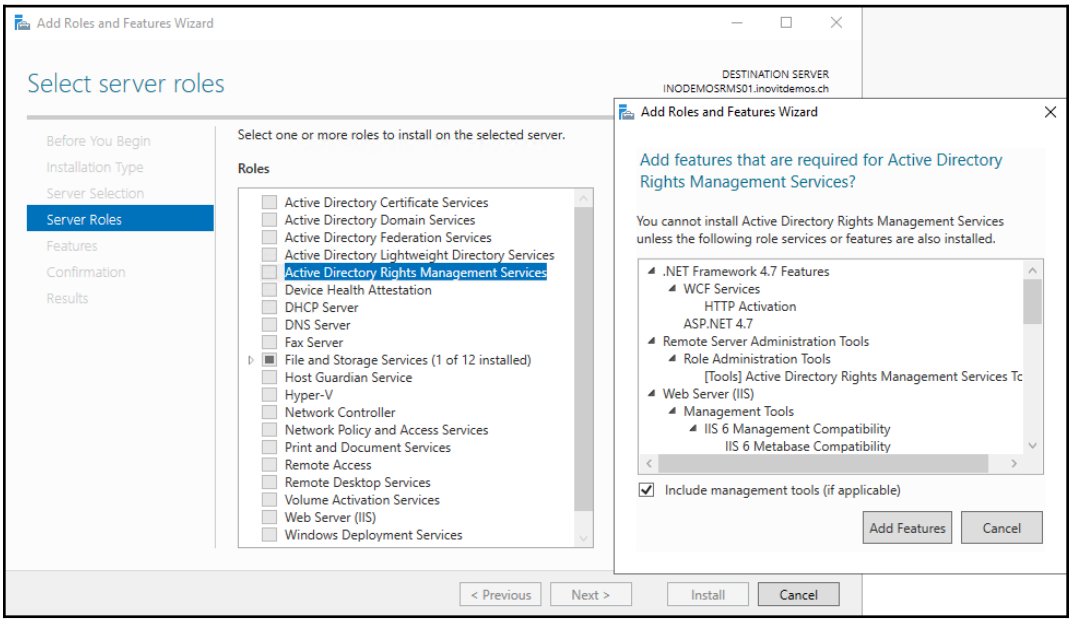
8. Create a test group such as `Executives`, for the RMS template and configure an email address—fill in the group with the users you want to test:

```
New-ADGroup -Name "Executives" -SamAccountName
"Executives" -GroupCategory Security -GroupScope Global -
DisplayName "Executives" -Path "OU=Groups,OU=Managed
Business Objects,DC=inovitdemos,DC=ch" -Description
"Executives"
```

9. Create an internal DNS entry for your RMS endpoints:

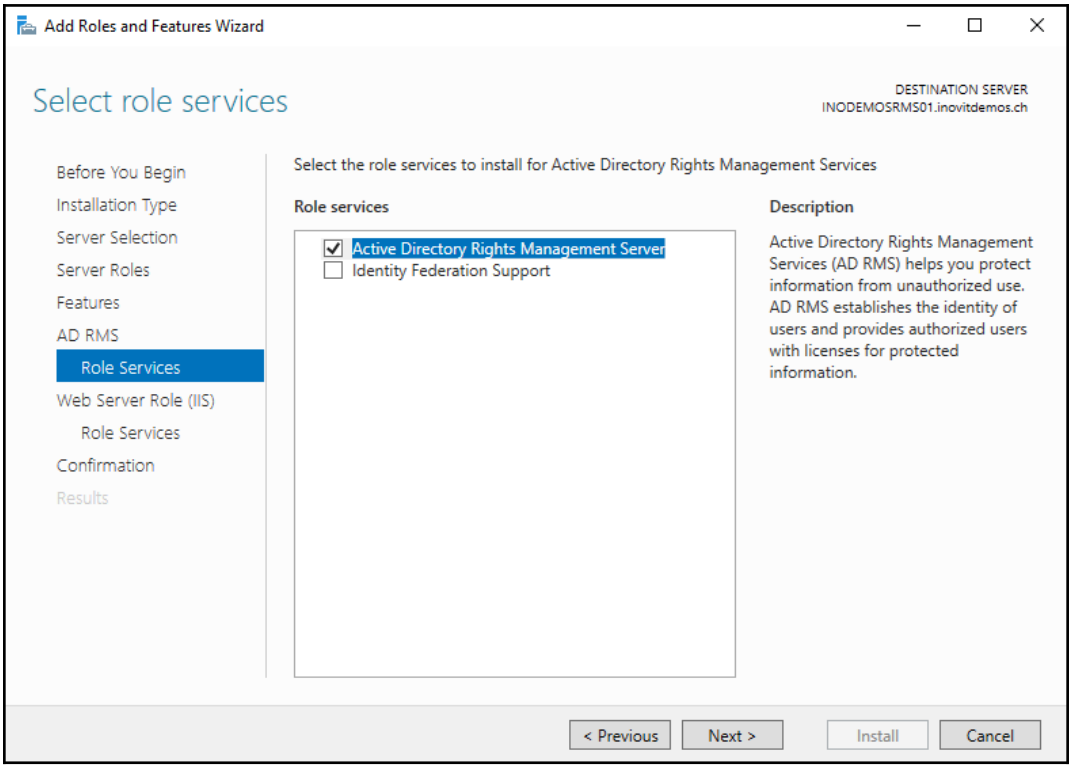
```
Add-DnsServerResourceRecord -ZoneName "inovitdemos.ch" -A
-Name "rms" -IPv4Address "10.0.0.11"
```

10. Log in as a domain administrator to the new server that's hosting the AD RMS server.
11. Open the server manager and add the **Active Directory Rights Management Services**:
    - Click **Add Features**:



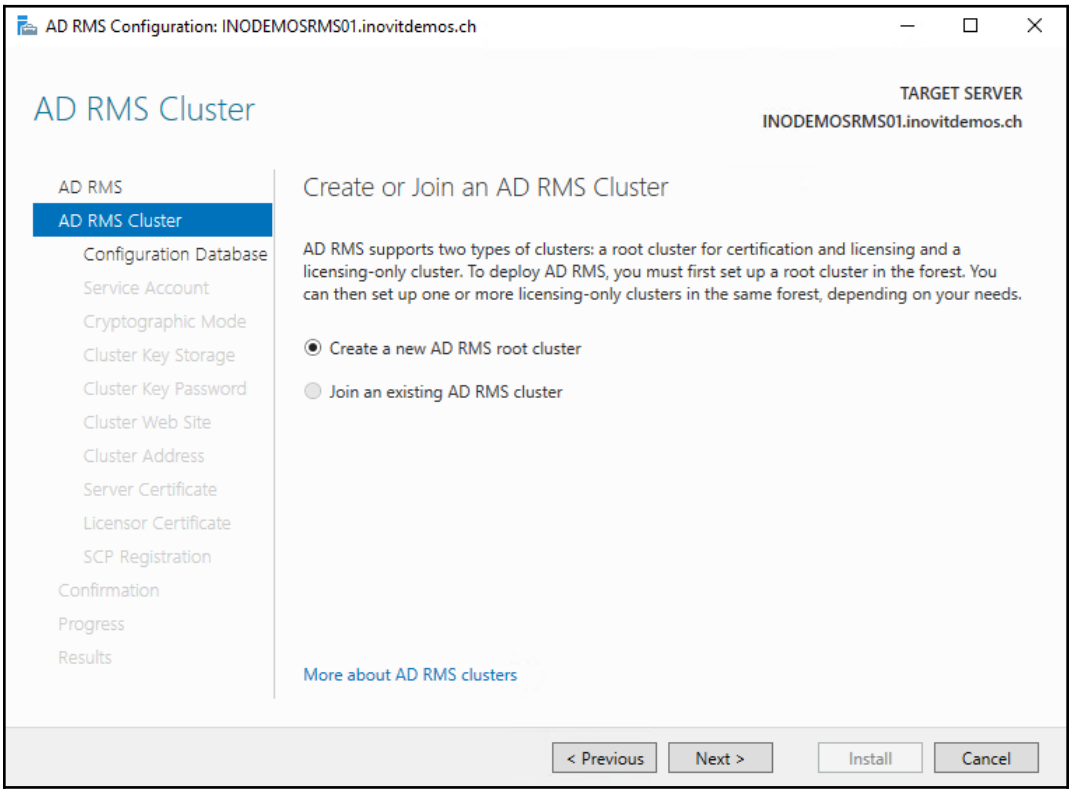
AD RMS role installation

12. Leave all the defaults as is and click **Next**:



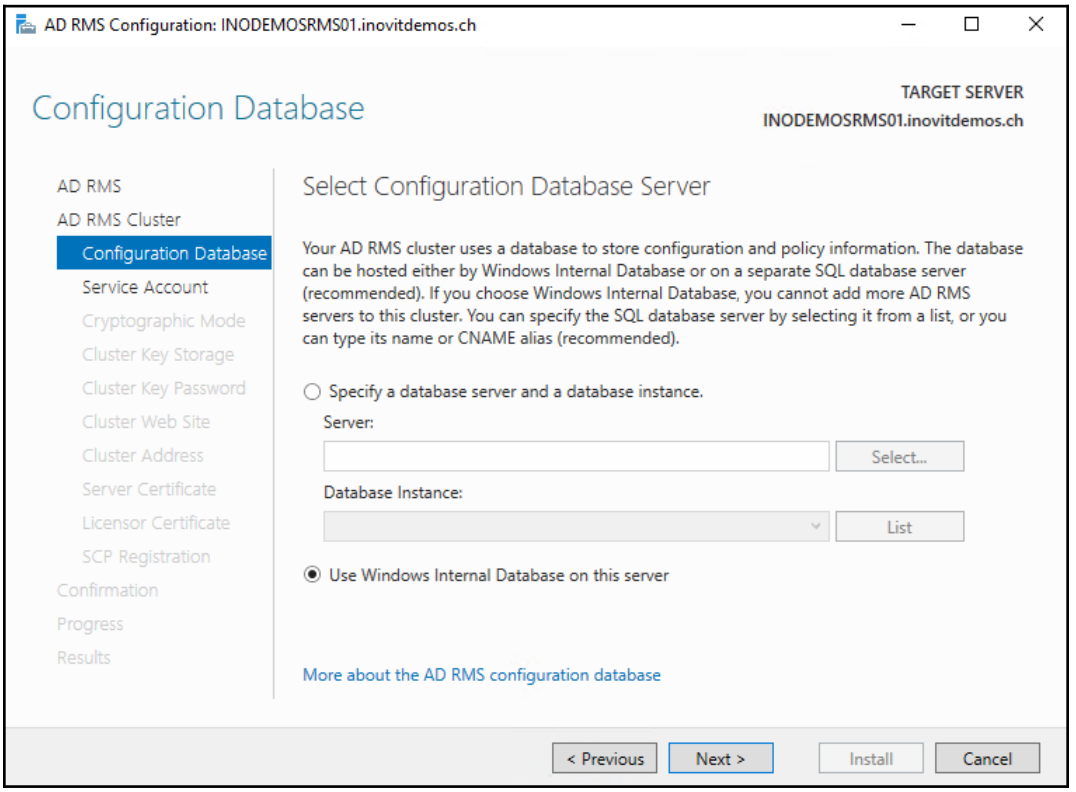
Role service installation

13. Configure AD RMS by clicking **Next** on the welcome screen.
14. Create the RMS cluster ("cluster" is a specific term in AD RMS that indicates that you can have more than one RMS server—it doesn't mean a Windows Server cluster):



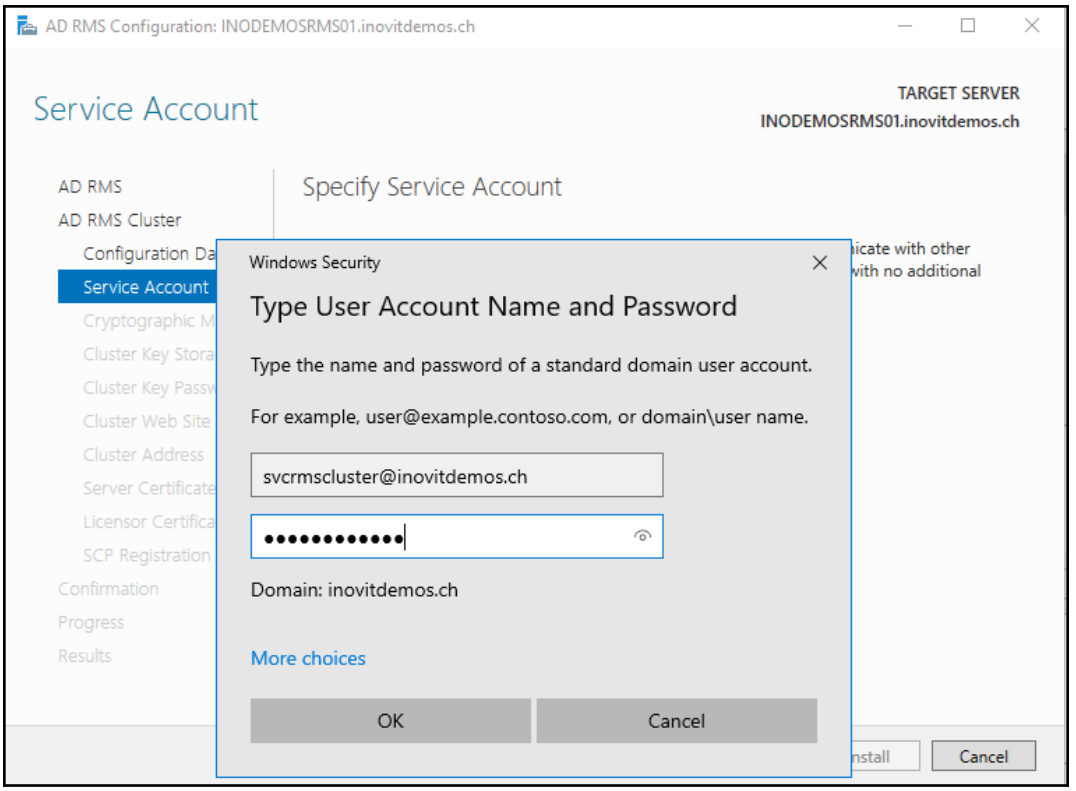
RMS cluster creation process

15. Choose the database deployment—for production environments, use a dedicated SQL server:



Database configuration for the AD RMS cluster

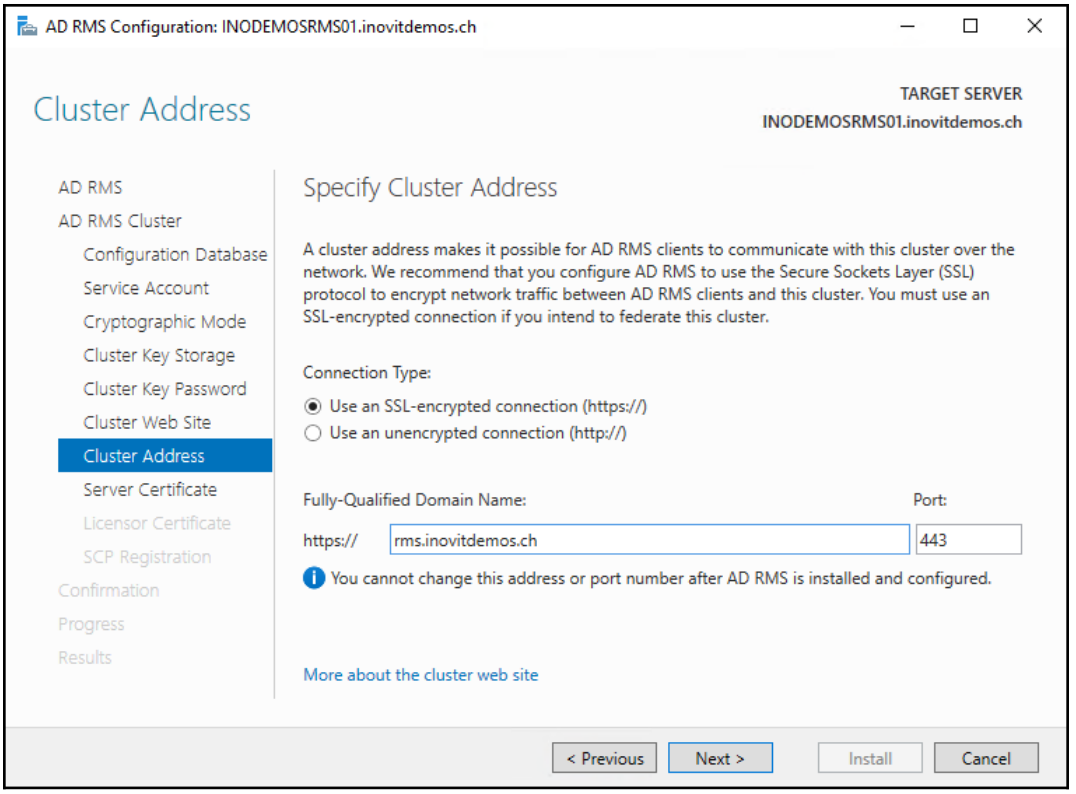
16. Configure the service account we created previously, that is, `svcrmscluster@inovitdemos.ch`:



Service account usage for AD RMS

17. Configure the following options, as follows:
    - **Cryptographic Mode**: Mode 2
    - **Cluster Key Storage**: Use AD RMS
    - **Cluster Key Password**: Choose your cluster key word like `455A#aasdd+!`
    - **Cluster Web Site**: Use the default website

18. Configure the cluster address:

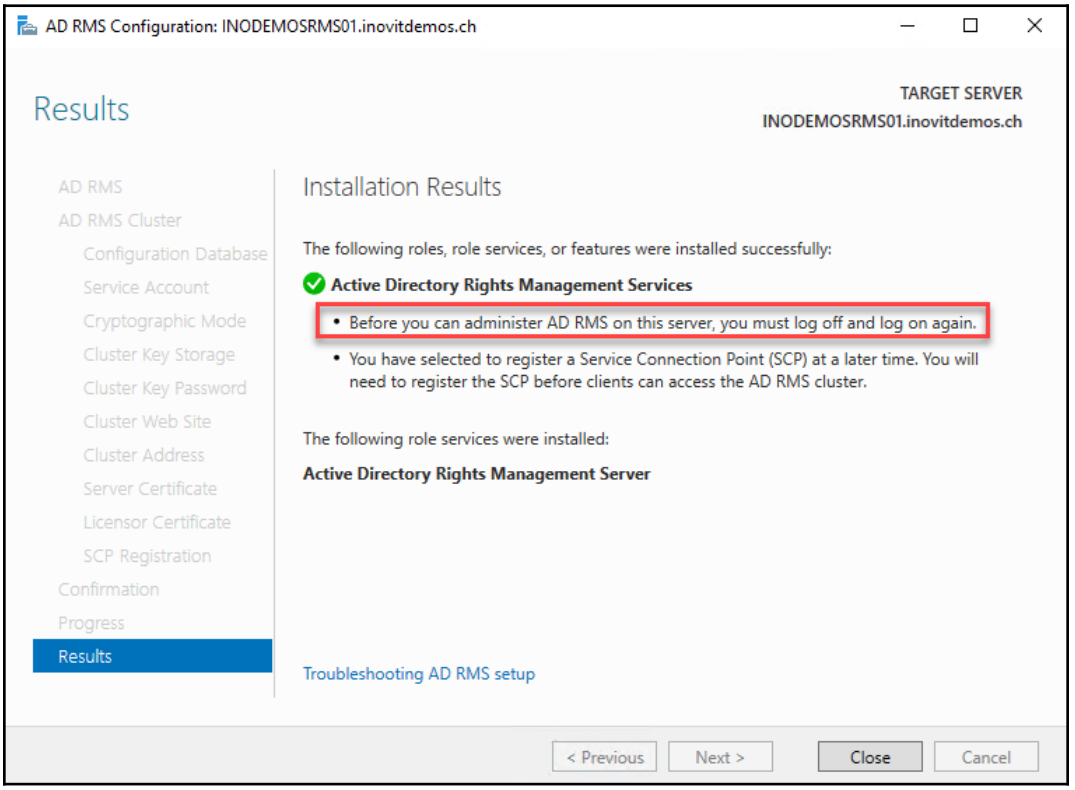


AD RMS Cluster address configuration

If you don't want to use HTTPS, change to HTTP for PoC/Demo purposes.

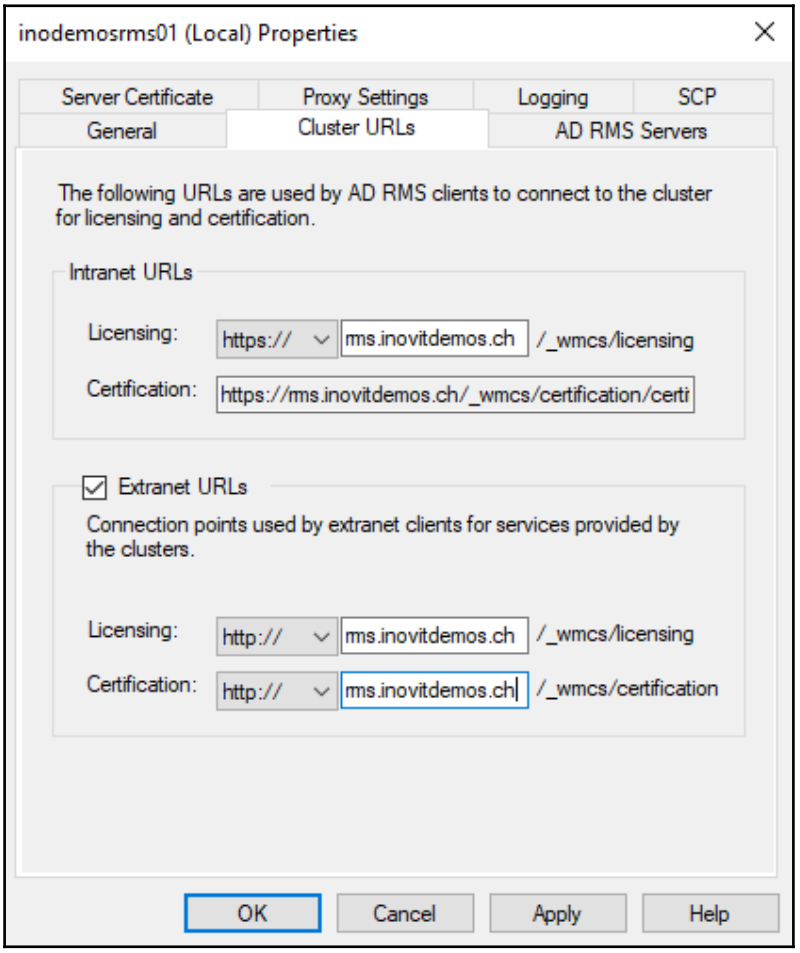
19. Choose your certificate if you configured HTTPS in the previous step.

20. Give your Licensor Certificate a name. We recommend using the organization name, such as `INOVITDEMOS`.
21. Don't register the SCP. If you're already on an existing infrastructure, you must change the option to not use an SCP.
22. We've almost finished the configuration of our AD RMS instance. We need to log out and log in again to complete the configuration:



AD RMS Installation result overview

23. Configure the last options by opening the AD RMS management console.
24. Open **Properties** and configure the **Extranet URLs**, like the **Intranet URLs**:

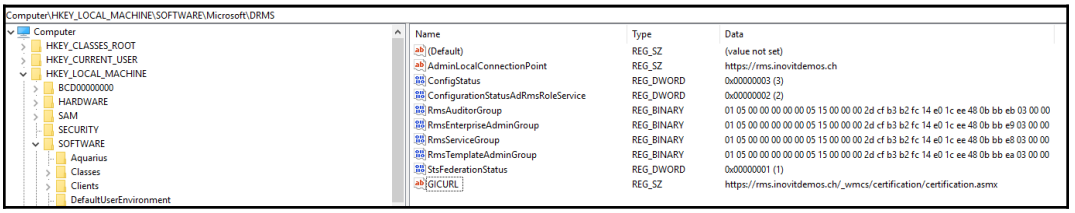


Configuring AD RMS Extranet URLs

25. Configure the AD RMS **Super User Group**:

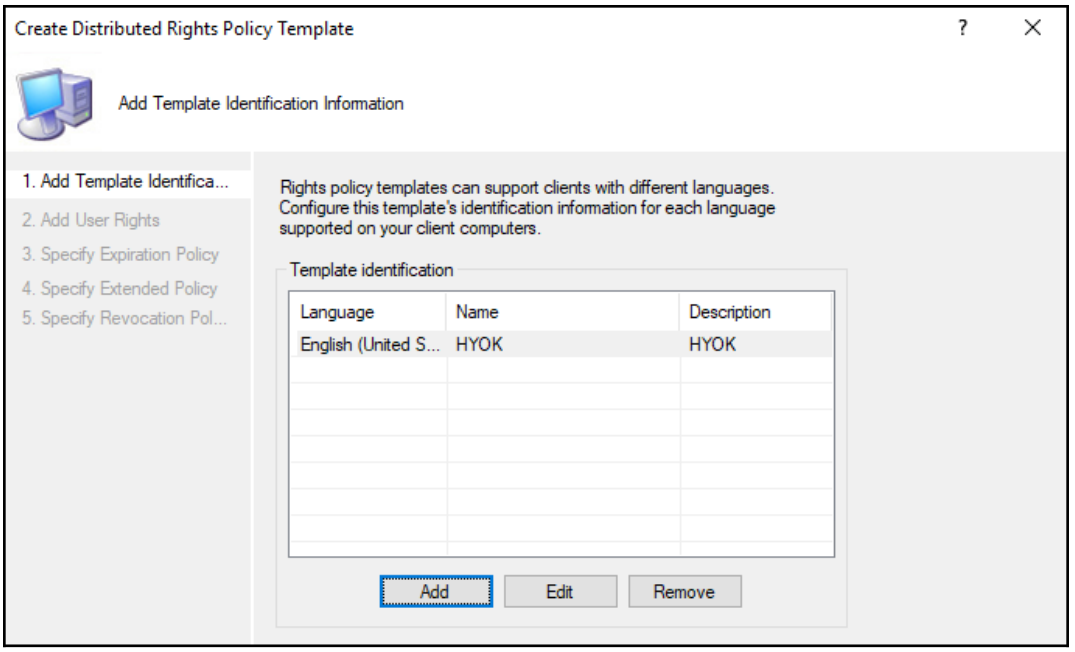
Providing the AD RMS Super User group

26. Open the registry editor to set the **GICURL REG_SZ** entry—`http or https rms cluster name /_wmcs/certification/certification.asmx`:



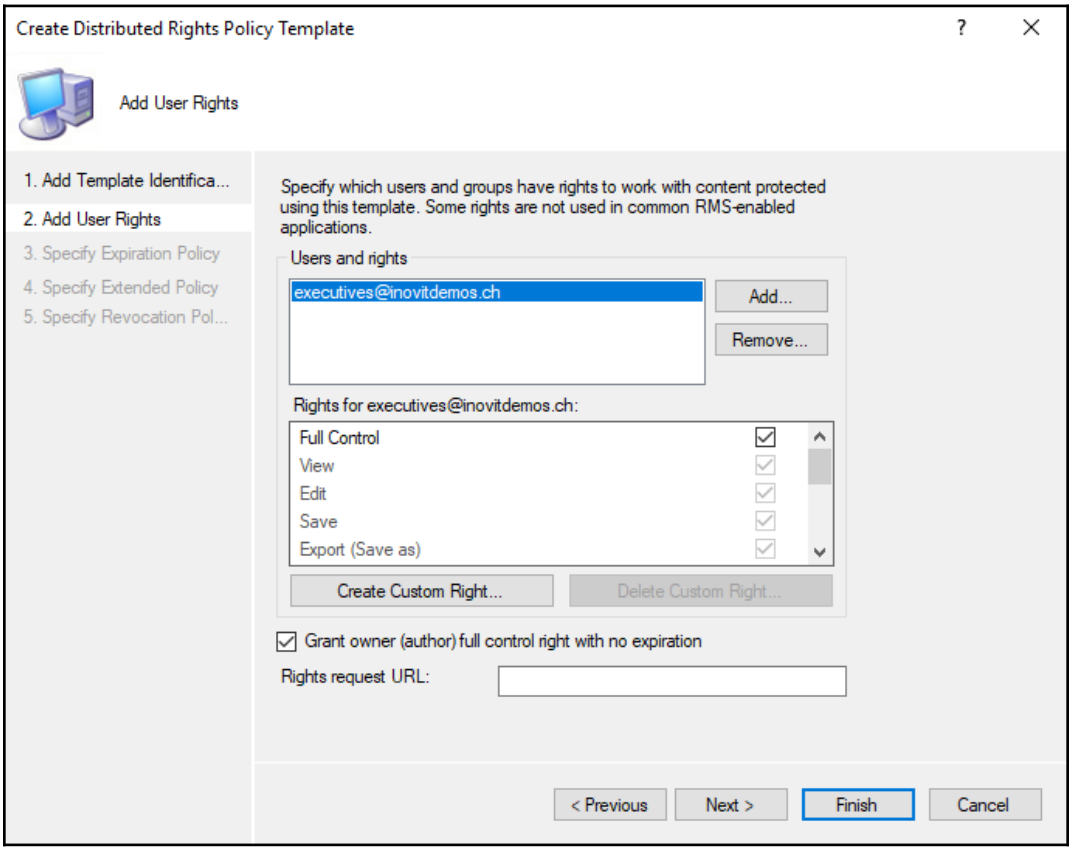
Setting the HYOK registry keys

27. Configure the RMS Template for HYOK:



HYOK RMS template settings

28. Define some example rights and the group you created previously to test its functionality:



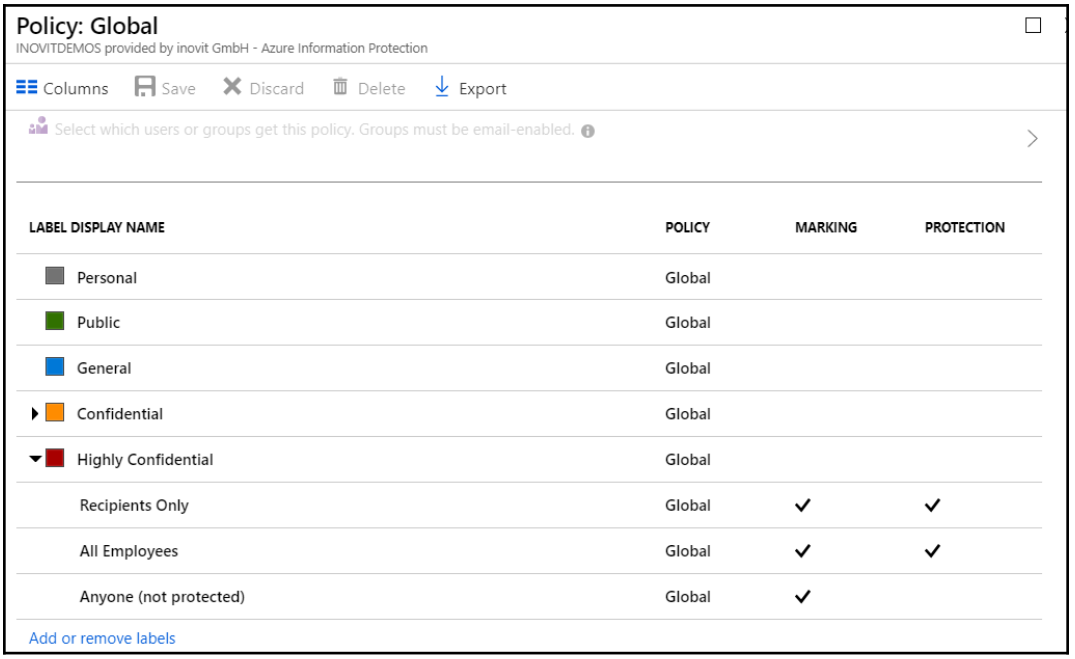
Assignment of the executive's group

29. Finish the configuration and open a PowerShell to gather the related information so that you can configure the Azure Information Protection label.

30. Run the following commands to get the RMS template GUID:

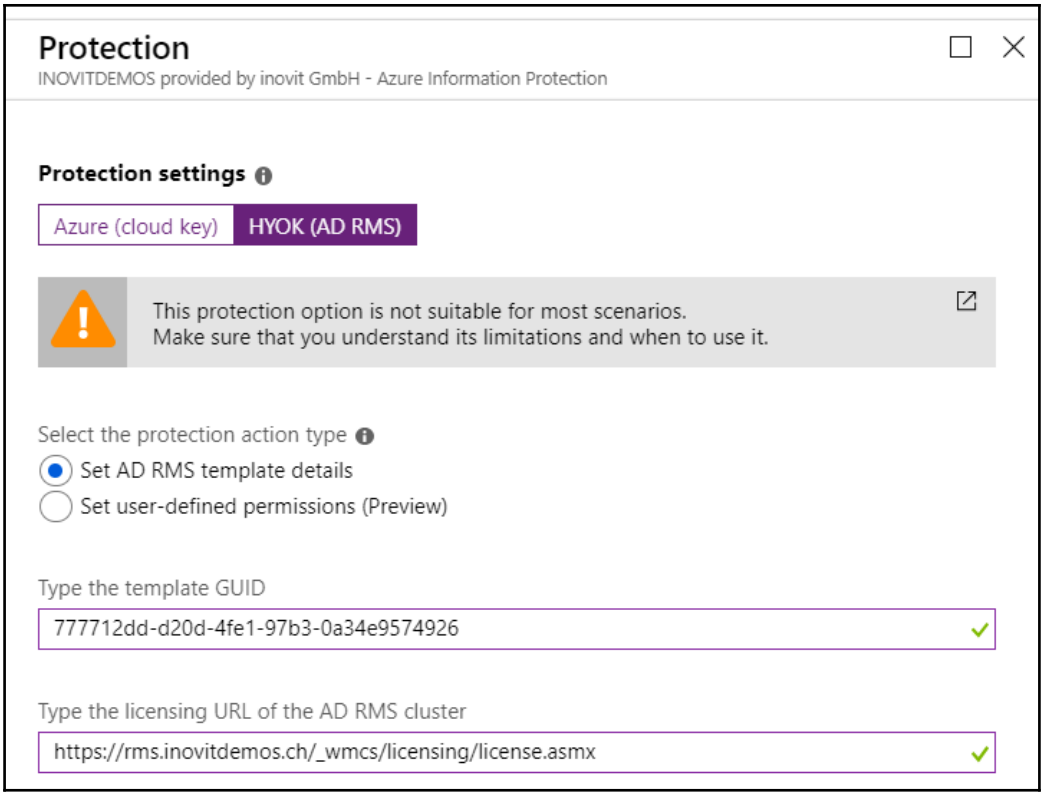
```
Import-Module AdRmsAdmin
New-PSDrive -Name AdrmsCluster -PSProvider AdRmsAdmin -
Root https://localhost
Get-ChildItem -Path AdRmsCluster:\RightsPolicyTemplate
```

31. Log into `https://portal.azure.com` to configure a new sub-label to test the HYOK configuration—add a new sub-label under **Highly Confidential**:



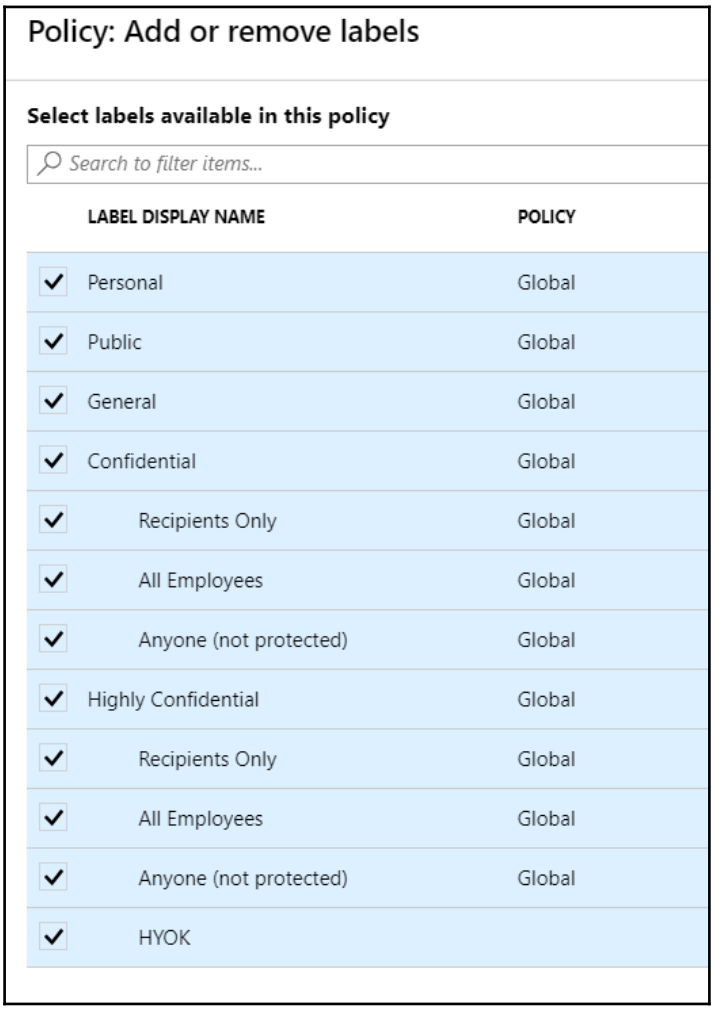
AIP Global Policy overview

32. Configure the label so:



Configuring the HYOK label

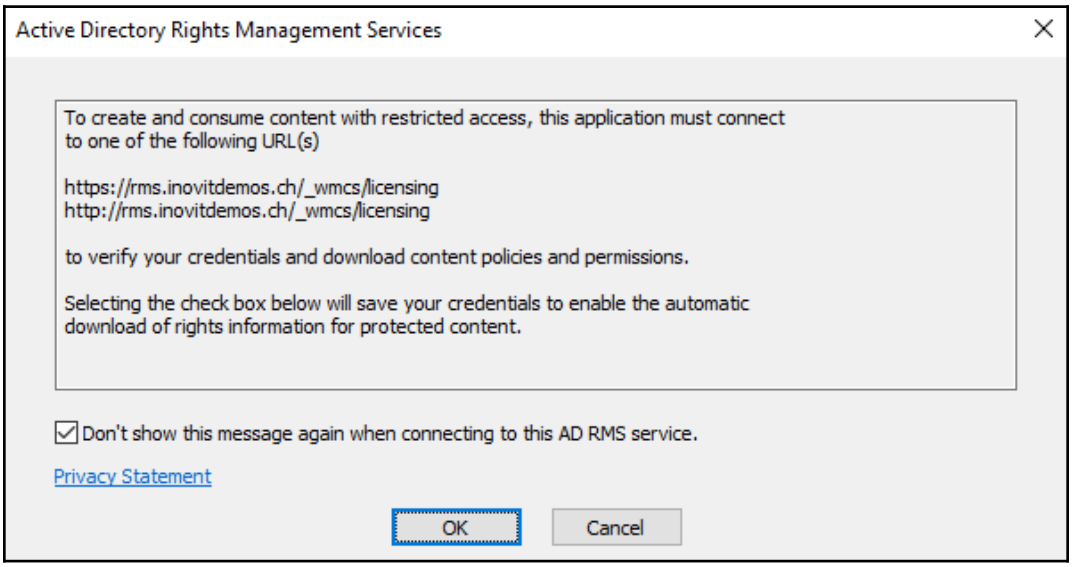
33. Enable the new sub-label for the global policy, which the test users will be able to choose:



Adding the label to the global policy

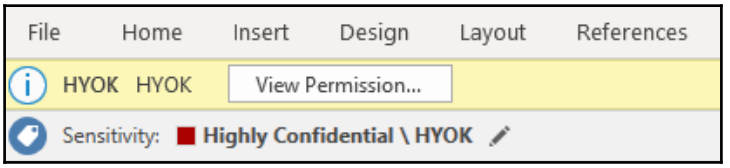
34. Open one of the Azure-Information-Protection-enabled internal domain-joined test clients from `Chapter 13`, *Identifying and Detecting Sensitive Data*.

35. Create a new document and choose the **HYOK** label. Save the document and you should get the following message the first time you use the new label:



First-time use message for the local RMS template

36. The document should be protected with the HYOK RMS template permissions:



HYOK protected document

You've successfully implemented an HYOK deployment in your test environment. Now that we've had a complete walk-through for all three deployment methods, we'll take a closer look at the Azure RMS functionality itself.

How Azure RMS works under the hood

It's quite important to how RMS works it delivers the data protection service of the complete Azure Information Protection solution. First of all, the protected data will never be transferred to the Azure Information Protection or Rights Management service itself. Basically, the data is encrypted at the application level and includes a policy that defines who is authorized and which usage rights are applied for these users. Keep in mind that group memberships are cached for three hours. So, if you change permissions, remember this to avoid misinterpreted results. To get a better understanding of the three typical flows, let's explore the flows in this section. But first, we'll look at the algorithms and key lengths used by Azure RMS.

Algorithms and key lengths

Azure RMS uses the following cryptographic controls in the different usage scenarios. The following table gives you the needed information if you get asked in a project or workshop:

Cryptographic controls	**Azure RMS usage**
Algorithm: AES **Key length**: 128/256 bits	Documentation protection **Used by**: • Azure Information Protection client • Rights Management sharing application
Algorithm: RSA **Key length**: 2,048 bits/*1,024 bits	Key protection • Migration from AD RMS running Cryptographic Mode 1 • Migration with AD RMS and Exchange Online usage • Archived keys (on-premises) before the migration • BYOK supports 1024/2048 bits - recommended: 2,048 bits
SHA-256	Certificate-signing

Now that we have a clear picture of the cryptographic controls, we can go further into the first use flow, or user environment-initialization.

User environment-initialization flow

The user environment-initialization is also called the bootstrapping process. The process starts when the Azure Information Protection client is installed on the client and a user opens an Office application, for example. It runs if a user consumes protected content or protects a newly-created document.

Keep in mind that if the user moves to another machine or another user uses the same machine, the process always runs the first time it's used.

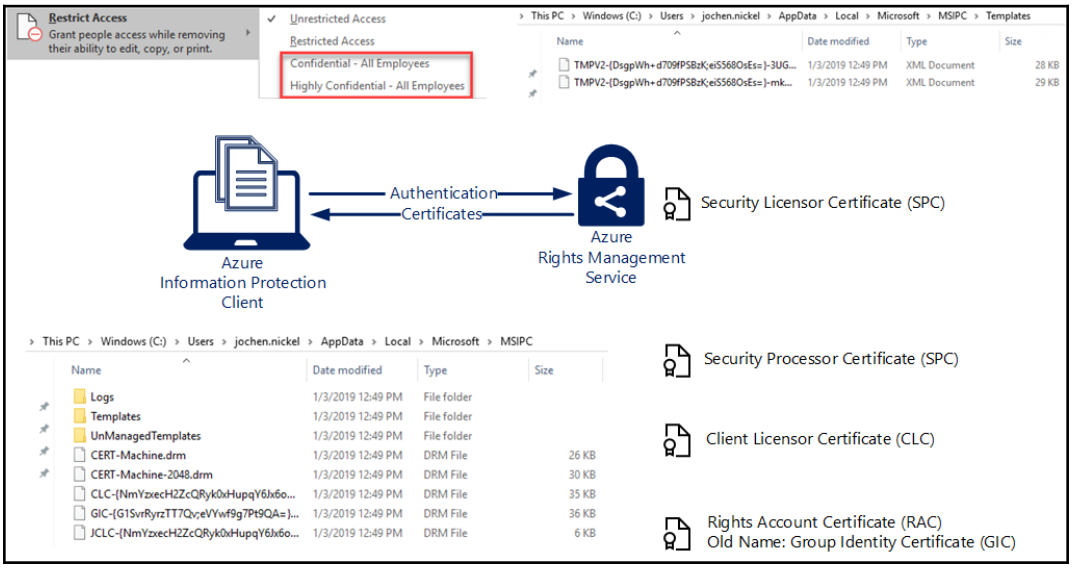
During this process, many things happen in the background, and if the user is provided by Single-Sign-On in a federated environment, they don't recognize the steps overall. After a successful authentication to the RMS service, the three main certificates will be received:

- The **Security Processor Certificate** (**SPC**)
- The **Client Licensor Certificate** (**CLC**)
- The **Rights Account Certificate** (**RAC**), formerly known as **Group Identity Certificate** (**GIC**)

These are all valid for 31 days and authenticate the user to the Azure Active Directory.

Also, the Rights Management templates from the organization will be received.

The following diagram shows the main components during this process:



Main components and actors on the bootstrapping process

Many other things happen during this process. If you need more details about the bootstrap process, you can find them at `https://bit.ly/2FbcxNt` and `https://bit.ly/2RxtcRR`.

Here are some other helpful sources:

- **AIP Client Admin Guide**: `https://bit.ly/2Txpoxr`
- ***The* Evolution of AD RMS to Azure Information Protection Part 6 by Matt Felton**: `https://bit.ly/2AwGR1Q`
- **Logging and analyzing usage**: `https://bit.ly/2RdJ3Wf`

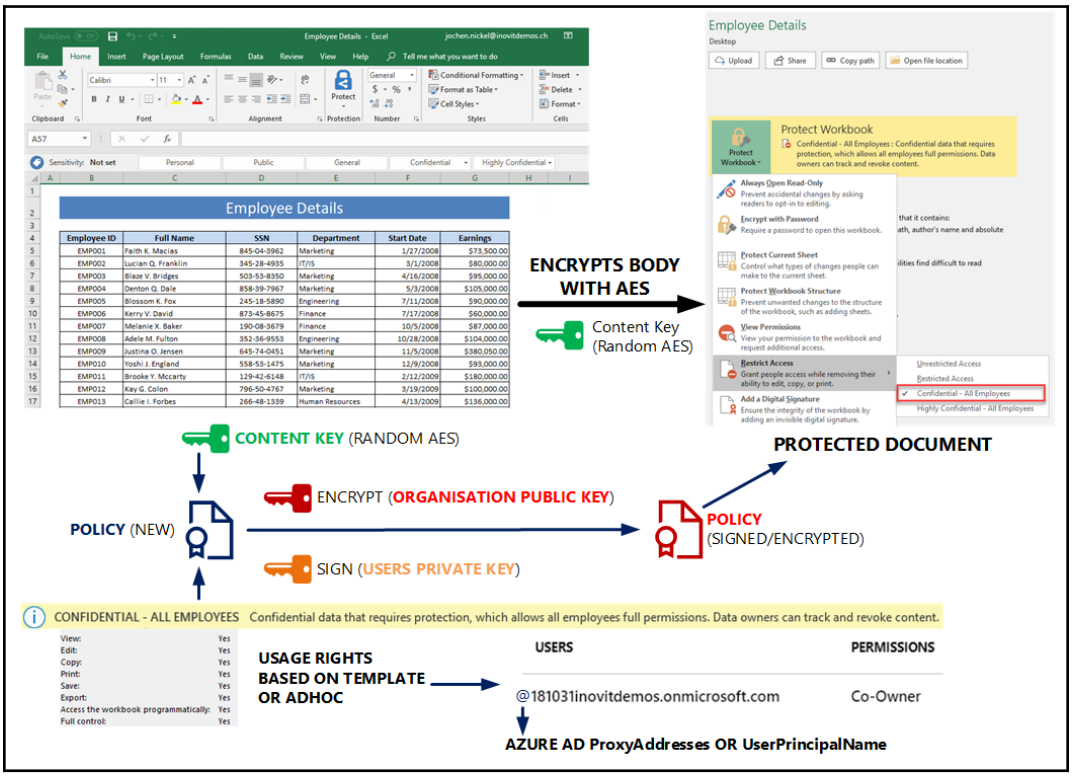
Next, we'll look into the content-protection flow.

Content-protection flow

The next is the content-protection flow, which is that happens if a user protects a document or email. The RMS client creates a random content key and encrypts the document using an AES symmetric-encryption algorithm. The flow is shown in full in the following diagram. Let's highlight just a few points:

- The main attribute to identify users or groups is the `ProxyAddresses` attribute; always keep old email addresses to be able to use older protected content
- If the `ProxyAddresses` are empty, `UserPrincipalName` will be used instead
- The RMS client uses the organization's key to encrypt the policy and the content key
- The RMS client signs the policy with the user's certificate
- The protection level stays always with the content

In the following diagram, you can see the content-protection flow:



Content-protection flow

Next, we will jump into the content-consumption flow.

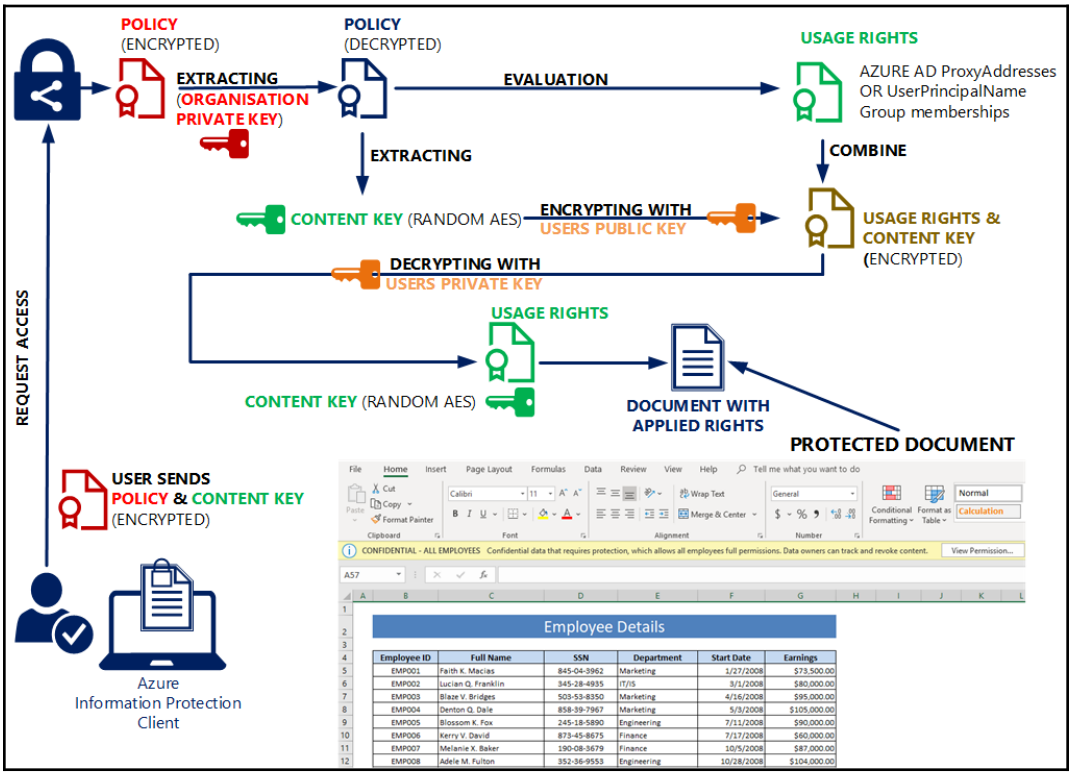
Content-consumption flow

Content consumption happens if a user tries to open an RMS-protected document or email. The process always starts by requesting access to the Azure Rights Management service. Let's highlight the most important points:

- The authenticated user sends the document policy and the user's certificates to the Azure Rights Management service
- The policy will be decrypted and evaluated by the service

- The service builds the rights for every user identification through the `ProxyAddresses` or `UserPrincipalName` attribute
- The content key gets extracted from the decrypted policy; the key will be encrypted with the user's public key
- The encrypted use license contains the re-encrypted content key
- The RMS client decrypts the encrypted use license with its own private key
- The RMS client decrypts the documents body and the rights list will be enforced by the application

The following view shows the content-consumption flow:



Content-consumption flow

With this flow, we've finished our journey through the Azure RMS flows. Well done!

Summary

In this chapter, you received all the necessary information to map your requirements to the correct key-deployment model. We looked at the pros and cons of using HYOK. With the provided configuration examples, we gathered practical experience, which you can now share or use in your next project or workshop. We learned about the main Azure RMS flows, which are helpful in understanding how Azure RMS works under the hood. You can use this knowledge to support deployments or troubleshoot common issues because many errors happen on the environment-initialization (bootstrapping) process or in the deployment.

In the next chapter, we'll finish configuring the example Azure Information Protection solution, and then you'll be fully prepared for your journey with this nice technology.

15
Configuring Azure Information Protection Solutions

After working through the theory and the detection and identification of sensitive information, we will work through some practical tips from our projects to help you to attain a better understanding of the technology and the associated processes. Always attempt to start projects with classification and without protection to avoid data loss and bad project marketing, because business processes don't work when users can't access the information they regularly use. In addition, always align training to your end users. This chapter will extend your lab environment and provide you with the important PowerShell cmdlets to administer your solution. Finally, we will configure **Azure Information Protection** (**AIP**) through some practical examples.

The chapter is divided into the following sections:

- Preparing to configure and manage AIP
- Azure RMS management with PowerShell
- Configuring AIP

Yeah! Let's start with the preparation tasks.

Preparing to configure and manage AIP

To configure and manage our AIP solution, we need to prepare the administrative Windows 10 workstation with the necessary tools. We need to have the following PowerShell modules installed on the workstation:

- Azure AD Preview: `Install-Module -Name AzureADPreview`
- Azure RMS: `Install-Module -Name AADRM`

Additionally, we install the AIP client on the machine from the following source: `https://bit.ly/2ccqSu0`.

For our use cases and the lab challenge, we need to create the required email-enabled groups. We choose to use Office 365 dynamic groups. You can use the following PowerShell cmdlets to create the required groups.

The first bunch of groups is required and the second one is optional:

```
# Connect to Azure AD and provide global administrator credentials
Connect-AzureAD

New-AzureADMSGroup -Description "Finance and Accounting Department
Users" -DisplayName "Finance and Accounting" -MailEnabled $true -
SecurityEnabled $true -MailNickname "financeandaccounting" -GroupTypes
"DynamicMembership","Unified" -MembershipRule "(user.department -
contains ""Accounting"")" -MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Sales Department Users" -DisplayName
"Sales" -MailEnabled $true -SecurityEnabled $true -MailNickname
"sales" -GroupTypes "DynamicMembership","Unified" -MembershipRule
"(user.department -eq ""Sales"") -or (user.department -eq ""Sales
Engagement Management"")" -MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Project Management Department Users"
-DisplayName "Project Management" -MailEnabled $true -SecurityEnabled
$true -MailNickname "projectmanagement" -GroupTypes
"DynamicMembership","Unified" -MembershipRule "(user.department -
contains ""Project Management"")" -MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Senior Management Department Users" -
DisplayName "Senior Management" -MailEnabled $true -SecurityEnabled
$true -MailNickname "seniormanagement" -GroupTypes
"DynamicMembership","Unified" -MembershipRule "(user.department -
contains ""Senior Management"")" -MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Sales Department Users" -DisplayName
```

```
"Sales" -MailEnabled $true -SecurityEnabled $true -MailNickname
"sales" -GroupTypes "DynamicMembership","Unified" -MembershipRule
"(user.department -eq ""Sales"") -or (user.department -eq ""Sales
Engagement Management"")" -MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Strategy Consulting Department Users"
-DisplayName "Strategy Consulting" -MailEnabled $true -SecurityEnabled
$true -MailNickname "strategyconsulting" -GroupTypes
"DynamicMembership","Unified" -MembershipRule "(user.department -
contains ""Strategy Consulting"")" -MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Human Resources Department Users" -
DisplayName "Human Resources" -MailEnabled $true -SecurityEnabled
$true -MailNickname "humanresources" -GroupTypes
"DynamicMembership","Unified" -MembershipRule "(user.department -
contains ""Human Resources"")" -MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Executives Department Users" -
DisplayName "Executives" -MailEnabled $true -SecurityEnabled $true -
MailNickname "executives" -GroupTypes "DynamicMembership","Unified" -
MembershipRule "(user.department -contains ""Executive"")" -
MembershipRuleProcessingState "On"
```

The following bunch of groups is optional:

```
New-AzureADMSGroup -Description "Project Management Department Users"
-DisplayName "Project Management" -MailEnabled $true -SecurityEnabled
$true -MailNickname "projectmanagement" -GroupTypes
"DynamicMembership","Unified" -MembershipRule "(user.department -
contains ""Project Management"")" -MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Operations Department Users" -
DisplayName "Operations" -MailEnabled $true -SecurityEnabled $true -
MailNickname "operations" -GroupTypes "DynamicMembership","Unified" -
MembershipRule "(user.department -eq ""Operations"") -or
(user.department -eq ""Engineering Operations"")" -
MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Marketing Department Users" -
DisplayName "Marketing" -MailEnabled $true -SecurityEnabled $true -
MailNickname "marketing" -GroupTypes "DynamicMembership","Unified" -
MembershipRule "(user.department -contains ""Marketing"")" -
MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Engineering Department Users" -
DisplayName "Engineering" -MailEnabled $true -SecurityEnabled $true -
MailNickname "engineering" -GroupTypes "DynamicMembership","Unified" -
MembershipRule "(user.department -contains ""Engineering"")" -
```

```
MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Contractors" -DisplayName
"Contractor" -MailEnabled $true -SecurityEnabled $true -MailNickname
"contractor" -GroupTypes "DynamicMembership","Unified" -MembershipRule
"(user.department -contains ""1099 Contractor"")" -
MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Content Management Consulting
Department Users" -DisplayName "Content Management Consulting" -
MailEnabled $true -SecurityEnabled $true -MailNickname
"contentmanagementconsulting" -GroupTypes
"DynamicMembership","Unified" -MembershipRule "(user.department -eq ""
Content Management Consulting"")" -MembershipRuleProcessingState "On"

New-AzureADMSGroup -Description "Customer Relationship Management
Department Users" -DisplayName "Customer Relationship Management" -
MailEnabled $true -SecurityEnabled $true -MailNickname
"customerrelationshipmanagement" -GroupTypes
"DynamicMembership","Unified" -MembershipRule "(user.department -
contains ""Customer Relationship Management"")" -
MembershipRuleProcessingState "On"
```

Furthermore, we need to set the correct mail suffix of the Office 365 groups. You can use the following commands to set the addresses:

```
# Set your preferred PowerShell execution policy
Set-ExecutionPolicy Unrestricted
# Provide global administrator rights
$UserCredential = Get-Credential
$Session = New-PSSession -ConfigurationName Microsoft.Exchange -
ConnectionUri https://outlook.office365.com/powershell-liveid/ -
Credential $UserCredential -Authentication Basic -AllowRedirection
Import-PSSession $Session
# Set the correct suffix
set-UnifiedGroup -Identity Sales -PrimarySmtpAddress
"sales@inovitdemos.ch"
set-UnifiedGroup -Identity Marketing -PrimarySmtpAddress
"marketing@inovitdemos.ch"
set-UnifiedGroup -Identity "Strategy Consulting" -PrimarySmtpAddress
"strategyconsulting@inovitdemos.ch"
set-UnifiedGroup -Identity "Project Management" -PrimarySmtpAddress
"projectmanagement@inovitdemos.ch"
set-UnifiedGroup -Identity Operations -PrimarySmtpAddress
"operations@inovitdemos.ch"
set-UnifiedGroup -Identity "Human Resources" -PrimarySmtpAddress
"humanresources@inovitdemos.ch"
set-UnifiedGroup -Identity Executives -PrimarySmtpAddress
```

```
"executives@inovitdemos.ch"
set-UnifiedGroup -Identity Engineering -PrimarySmtpAddress
"engineering@inovitdemos.ch"
set-UnifiedGroup -Identity Contractor -PrimarySmtpAddress
"contractor@inovitdemos.ch"
set-UnifiedGroup -Identity "Finance and Accounting" -
PrimarySmtpAddress "financeandaccounting@inovitdemos.ch"
set-UnifiedGroup -Identity "Customer Relationship Management" -
PrimarySmtpAddress "customerrelationshipmanagement@inovitdemos.ch"
set-UnifiedGroup -Identity "Content Management Consulting" -
PrimarySmtpAddress "contentmanagementconsulting@inovitdemos.ch"
set-UnifiedGroup -Identity "Senior Management" -PrimarySmtpAddress
"seniormanagement@inovitdemos.ch"
```

Be aware that this is a list of users that are fully equipped with a Microsoft 365 E5 license:

Name	Department	Job Title	Manager
Dan Jump	Executive	CEO	
Adam Barr	Operations	General Manager of Professional Services	Dan Jump
Karim Manar	Accounting	Controller	Diane Tibbot
Dan Park	Sales	Vice President NA Sales	Adam Barr
Christa Geller	Executive	CVP of Online	Jeff Hay
Alan Brewer	Sales	Regional Sales Manager	Jack Creasey
Aaron Painter	Strategy Consulting	Strategy Consulting Manager	Christine Koch
Amy Alberts	Human Resources	HR Manager	Garth Fort
Scott Bishop	Project Management	Senior Project Manager	Alan Steiner
Don Hall	Strategy Consulting	Strategy Consultant	Ellen Adams
Chase Carpenter	Accounting	Accountant	Karim Manar
Lars Hansson	Accounting	Accounting Manager	Karim Manar
David Wright	Sales	Salesperson	David Simpson
Greg Winston	Senior Management	President of Management	Jeff Hay
Ye Xu	Sales Engagement Management	Salesperson	Arthur Yasinski
Ian Tien	Human Resources	HR Specialist	Amy Alberts
Brian Cox	Operations	Procurement Manager	Keith Dishmo

We have prepared our workstation and created the related use cases resources. Next, we will discuss the most important PowerShell cmdlets to configure and manage the Azure RMS service.

Azure RMS management with PowerShell

In the following section, we will discuss and use the most important PowerShell commands to configure and manage Azure RMS. You will get a good overview of the features and how to start the configuration.

Azure RMS super users

We will start with the connection to the Azure RMS service and the management of the super users feature. This feature is disabled by default. To use its functionality, we need to enable the feature and assign a mail-enabled group to it. We highly recommend adding the AIP Scanner account permanently to the group, and all the other required users as and when required. It doesn't matter when the user is added to decrypt information from the past.

The Azure RMS super users feature provides the following functionality in Azure RMS:

- Full control over all rights-protected content that is managed by rights management
- Full owner rights to super users for all user licenses that are issued by the subscriber's organization
- Decryption of any rights-protected content file and removal of rights-protection from it for content previously protected within that organization

Typical use cases for the super user feature are:

- Read-protected information from an employee that leaves the organization
- Replacement of a currently assigned protection policy
- Usage of search operations in Exchange Online (eDiscovery)
- Support for security solutions that need to inspect protected information

With the following steps, you can configure the super users feature:

1. Connect to the Azure RMS service with global administrator rights, as follows:

```
Connect-AadrmService
```

2. Get all super user commands, as follows:

```
Get-Command "*SuperUser*"
```

3. Check the super user feature, as follows:

```
Get-AadrmSuperUserFeature
```

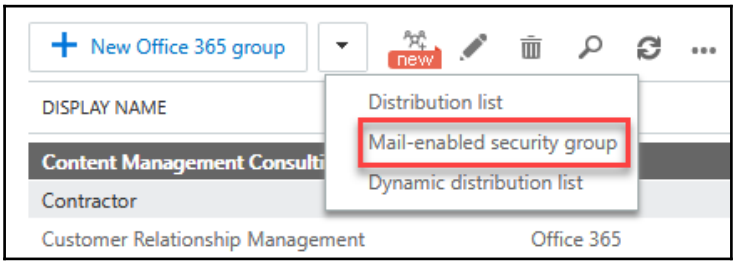
4. Enable the super user feature, as follows:

```
Enable-AadrmSuperUserFeature
```

5. Get all super users, as follows:

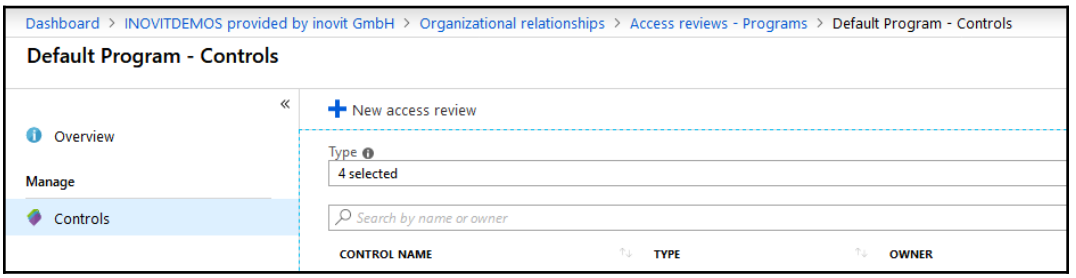
```
Get-AadrmSuperUser
Get-AadrmSuperUserGroup
```

6. Next, we create a **Mail-enabled security group** in the **Exchange Online** administrative center.
7. Navigate to **Recipients** | **Groups** and create a new **Mail-enabled security group**, as follows:



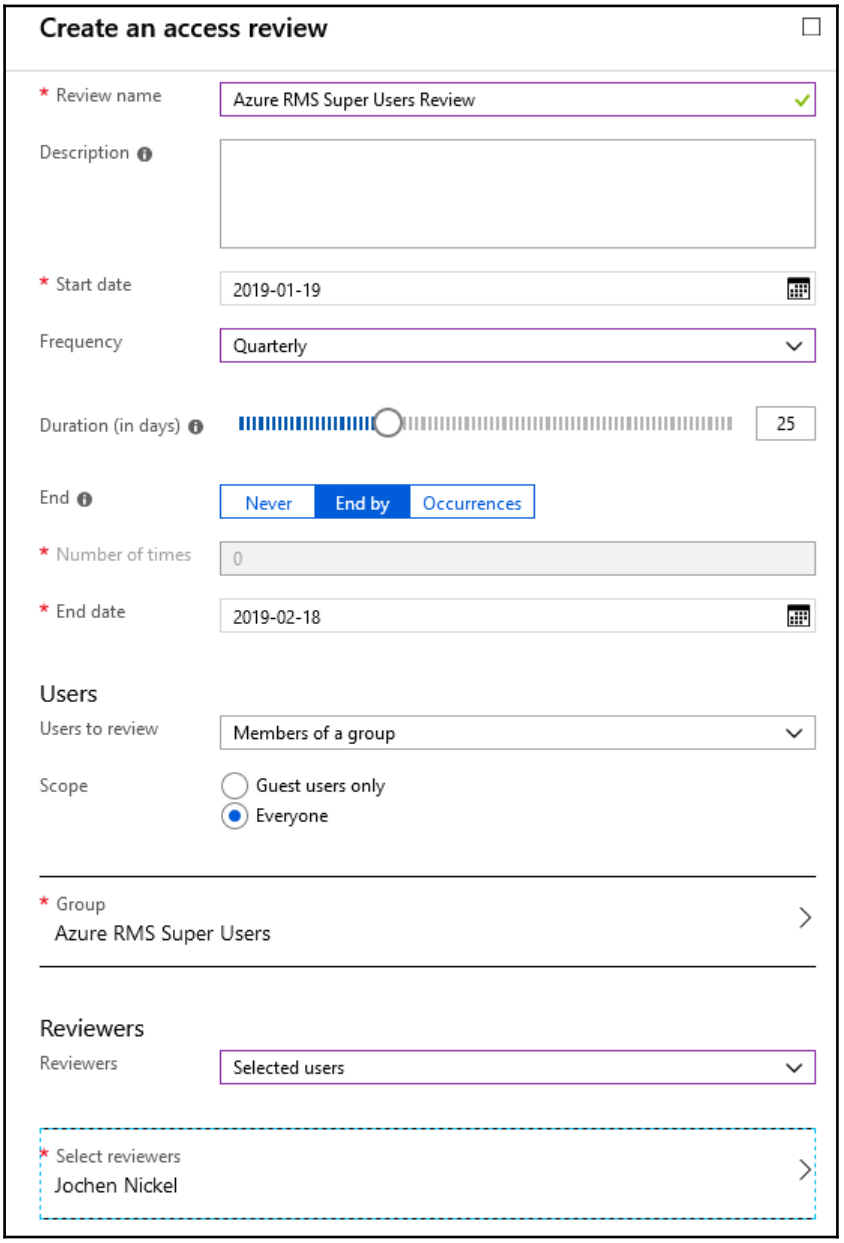
Mail-enabled security group creation

8. After we create the group, we will also use **Access reviews** to review the membership quarterly of the group, because the group is very powerful and shouldn't be filled with old users.
9. Open the Azure portal at `https://portal.azure.com` and navigate to the Azure AD blade.
10. Choose **Organizational relationships** | **Access reviews**.
11. Click **Onboard** and **Create** for the actual Azure AD.
12. Use the **Default Program** and under **Manage** | **Controls**.
13. Click **New access review**, as follows:



Access review creation

14. Choose the following settings for the review and assign your personal test account as a reviewer:



Access review properties

15. Next, we will assign the group as super users group.Set a super user group, as follows:

```
Set-AadrmSuperUserGroup -GroupEmailAddress
"azurermssuperusers@inovitdemos.ch"
```

Now, we are safe if the organization starts encrypting information. All the protected content can be unprotected and recovered by the users in the super user group.

To administer the Azure RMS service, we can delegate administrative rights with the following cmdlets:

```
Get-AadrmRoleBasedAdministrator
Add-AadrmRoleBasedAdministrator
```

If you use a group for this delegation, the group does not need to be mail-enabled.

On the other hand, you should assign the administration for AIP in the Azure AD **Privileged Identity Management** (**PIM**) blade, as follows:

Information Protection Administrator - description on Azure AD

In the next section, we will discuss onboarding controls.

Onboarding controls

It's recommended to do a planned roll out of AIP, including the RMS protection functionality in an organization. For this reason, Microsoft has implemented the onboarding controls feature. You can check the default configuration with the following command:

```
Get-AadrmOnboardingControlPolicy
```

Open an elevated PowerShell after you have established a connection to the Azure RMS service with the following command:

```
Connect-AadrmService
```

The feature is disabled by default.

To enable onboarding controls, you can use the following command:

```
Set-AadrmOnboardingControlPolicy
```

With onboarding controls, you can enable a group-based rollout. To be that the users and associated computers are already prepared to use AIP. To prepare your users, you can use the following source: `https://bit.ly/2CvBfF7`.

Azure RMS templates

Azure RMS templates are basically used over labels in AIP. If an organization doesn't use AIP for classification and labeling information, they can still use the basic Azure RMS service to protect content. Azure RMS templates can be created over the AIP blade under a label or by PowerShell. The actual Azure RMS usage rights are documented at `https://bit.ly/2HkDK2F`. To manage Azure RMS templates, we can use the following procedures:

1. Get all template commands, as follows:

```
Get-Command -Module AADRM Template
```

2. Get all templates, as follows:

```
Get-AadrmTemplate
```

The output of the preceding command is as follows:

```
PS C:\> Get-AadrmTemplate


TemplateId  : 026be843-becf-425f-a776-27d4c1b8fd54
Name        : Confidential - All Employees
Description : Confidential data that requires protection, which allows all employees full permissions. Data owners can
              track and revoke content.

TemplateId  : 68ab2d70-f65e-4b83-b3fb-8ed3e7f06ee7
Name        : Highly Confidential - All Employees
Description : Highly confidential data that allows all employees view, edit, and reply permissions to this content.
              Data owners can track and revoke content.
```

Get all actual RMS templates

3. Remove templates, as follows:

```
Remove-AadrmTemplate -TemplateId "template ID"
```

4. Create a new template, as follows:

```
$names = @{}
 $names[1033] = "Template Name"
 $descriptions = @{}
 $descriptions[1033] = "Template description"
 $r1 = New-AadrmRightsDefinition -DomainName "yourdomain1.com"
-Rights "VIEW","EXPORT"
 $r2 = New-AadrmRightsDefinition -EmailAddress "Email address
of group or user" -Rights "OWNER"
 Add-AadrmTemplate -Names $names -Descriptions $Descriptions -
LicenseValidityDuration 7 -RightsDefinitions $r1, $r2 -Status
Published
```

In the next section, we will provide some important information about the Azure RMS logging features.

Azure RMS logging

In this section, we will verify the default enabled Azure RMS logging. Perform the following steps to analyze the usage of the Azure RMS service:

1. Get the actual Azure RMS configuration, as follows:

```
Get-AadrmConfiguration
```

2. Get all the log commands, as follows:

```
Get-Command "*Log*" -Module AADRM
```

Create an AIP log directory and set the variables, as follows:

```
New-Item -ItemType directory -Path C:\AIPLogs\User
New-Item -ItemType directory -Path C:\AIPLogs\Admin
$UserLogs = "C:\AIPLogs\User"
$AdminLogs = "C:\AIPLogs\Admin"
```

3. Get the user logs, as follows:

```
Get-AadrmUserLog -Path $UserLogs -FromDate (Get-
Date).AddDays(-45)
```

4. Get the admin logs, as follows:

```
Get-AadrmAdminLog -Path $AdminLogs\admin.log
```

5. Get the status from the document tracking feature, as follows:

```
Get-AadrmDocumentTrackingFeature
```

The next section will take you through an introduction to the AIP PowerShell capabilities.

AIP client PowerShell

With the AIP client installed, you can gather information about your configuration and do classification and protection activities. You can use the following command with the AIP cmdlets:

```
Get-Command -Module AzureInformationProtection
```

With the following cmdlet, you get the actual available RMS templates on a client:

```
Get-RMSTemplate
```

You can get the actual state of a file with the following cmdlet:

```
Get-AIPFileStatus .\Q3_Product_Strategy.docx
```

```
PS C:\Users\donh\Desktop> Get-AIPFileStatus .\Q3_Product_Strategy.docx


FileName        : C:\Users\donh\Desktop\Q3_Product_Strategy.docx
IsLabeled       : True
MainLabelId     : de82bccd-c50f-4162-b113-8aa9e98ed45f
MainLabelName   : Confidential
SubLabelId      : 6eae6a7b-f321-4fc4-8049-1ef7cc9575b2
SubLabelName    : All Employees
LabelingSiteId  : 7709ca2b-3be8-4d92-89d7-dc1e274b4d0e
Owner           : Don.Hall@inovitdemos.ch
LabelingMethod  : Manual
LabelDate       : 1/18/2019 10:20:31 AM
IsRMSProtected  : True
RMSTemplateId   : 026be843-becf-425f-a776-27d4c1b8fd54
RMSTemplateName : Confidential - All Employees
RMSIssuedTime   : 1/18/2019 10:21:00 AM
RMSOwner        : Don.Hall@inovitdemos.ch
RMSIssuer       : Don.Hall@inovitdemos.ch
```

AIP file status information

You can set a label with the following cmdlet:

```
Set-AIPFileLabel -LabelId de82bccd-c50f-4162-b113-8aa9e98ed45f -Path
.\Testfile.docx
```

After this administrative introduction section, we will follow up with different use cases to configure and use AIP.

Configuring AIP

Configuring and managing AIP should always start with the global approach for all users. The most important task to do before you start touching the technology is to work on a clear classification schema and the associated policies. Doing the configuration is the smallest part in a classification and information project. You should consider the default labels based on your classification schema, and start without encryption and a lot of automatic classification rules. Start with the global users that understand and work with the new technology. Keep in mind to use a step-by-step approach and work through the specific requirements in the most sensitive departments, such as human resources, legal, or finance. Don't overwhelm your users and don't teach them to lie to your concept and classification system.

Here are some additional tips for data classification:

- Gather the support of the management and employees who will use the system
- Tagging and categorizing everything is nearly impossible
- Use an effective metadata strategy
- Use data cleansing technology to remove redundant or obsolete data
- Provide availability
- Consider the confidentiality and security of the data to be classified

Microsoft has carried out some intensive research and usability testing on the labels used to apply your classification to the data. Try to start with the default labels of AIP, for example:

AIP default label set

You can also customize every label in AIP to fulfill your needs.

Creating the classification schema

The first step in configuring a AIP solution is to start with the creation of the classification schema. Let's create our classification schema for our demo organization in the following steps:

1. Open the AIP blade at `https://portal.azure.com`.

From `Chapter 14`, *Understanding Encryption Key Management Strategies*, we have an enabled HYOK label. We will remove this label for now, because we don't want to include the on-premises part in our solution.

2. The default schema should look like the following:

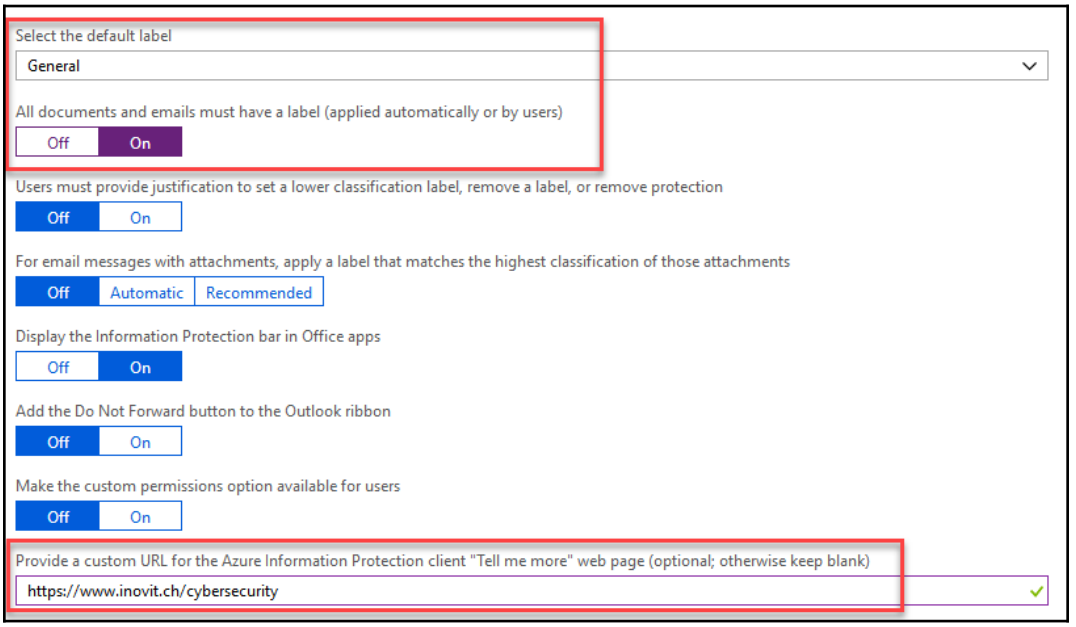
LABEL DISPLAY NAME	POLICY	MARKING	PROTECTION	
Personal	Global			...
Public	Global			...
General	Global			...
Confidential	Global			...
Recipients Only	Global	✓	✓	...
All Employees	Global	✓	✓	...
Anyone (not protected)	Global	✓		...
Highly Confidential	Global			...
Recipients Only	Global	✓	✓	...
All Employees	Global	✓	✓	...
Anyone (not protected)	Global	✓		...
+ Add a new label				

Default classification/label schema of AIP

A good starting point is that you start labeling your information with a default label and disable the encryption functionality for the **Confidential** and **Highly Confidential** labels.

3. You find the configuration under **Policy** | **Global**.

4. Use **General** as default label and ensure that all the information is required to be classified for your global users, for example:



Default label activation and other global settings

We highly recommend that you provide your users with an additional source for more information about data classification. In our projects, we use a very often used website for the users.

5. Set the labels that have protection enabled to **Not configured**, as follows:

Protection disabled option

6. Test the result on one of the test clients, as Don Hall.

7. A new document or email gets automatically classified as **General**, as follows:

Default labeling result

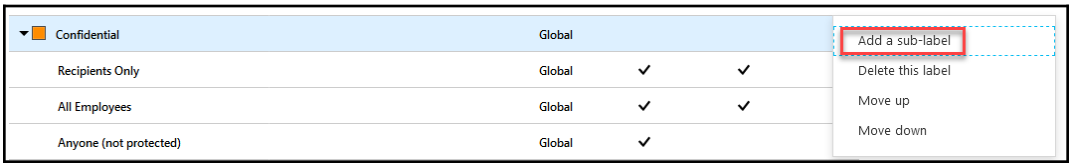
Business data under **General** label is not intended for public consumption. However, this can be shared with external partners as required. Examples include a company's internal telephone directory, organizational charts, internal standards, and most internal communications.

Creating sub-labels and scoped policies

Now that we have provided the first set to our users, we can start to provide a basic configuration for our sensitive departments. We can do that with scoped policies to provide the changes only to specific department users. Scoped policies require AIP P2 licenses.

With the following steps, we will create the required sub-labels:

1. Under **Classifications** | **Labels**. Create a new sub-label under the **Confidential** label, as follows:



Adding a sub-label

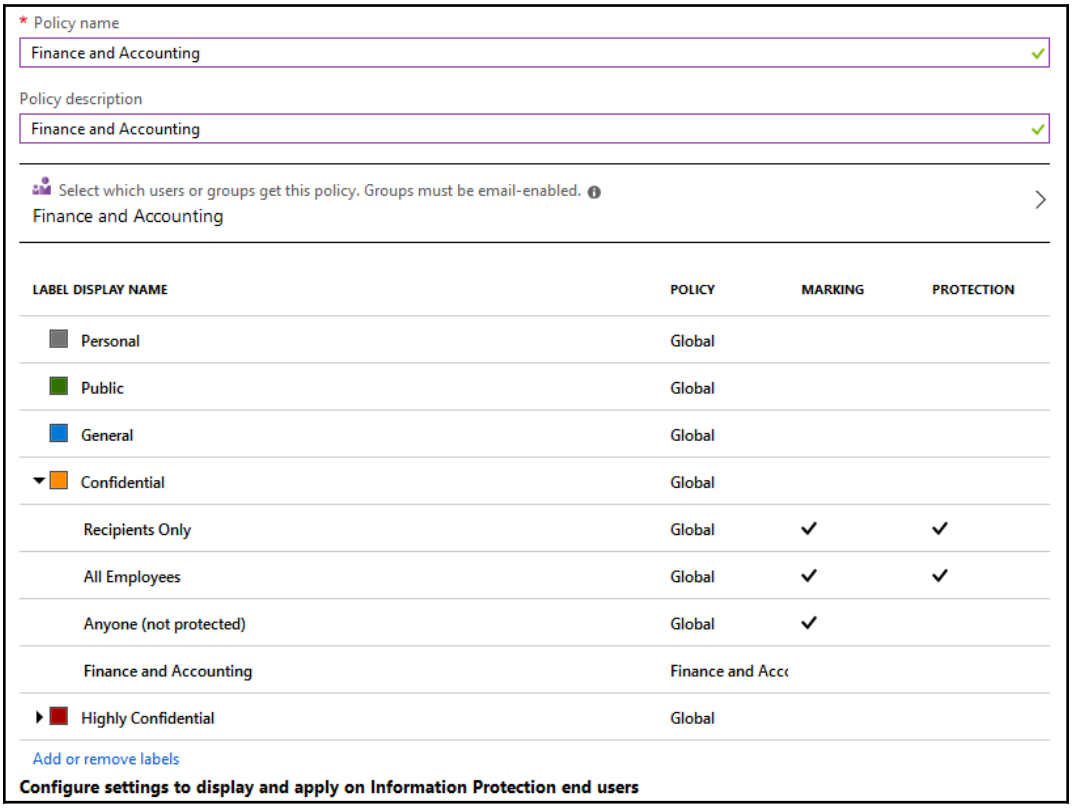
- **Name**: `Finance and Accounting`
- **Description**: `Finance and Accounting` sensitive data that is not intended for consumption outside the department
- Add the following conditions: credit card number and **International Banking Account Name** (**IBAN**)
- Leave **Recommended** to apply the label

2. Next, we will configure the custom policy for the **Finance and Accounting** department.

3. Under **Classifications** | **Policies**. Create one policy for `Finance and Accounting`.
4. Assign the `Finance and Accounting` group to the policy.
5. Click **Add or remove labels**.
6. Add the **Finance and Accounting** label.

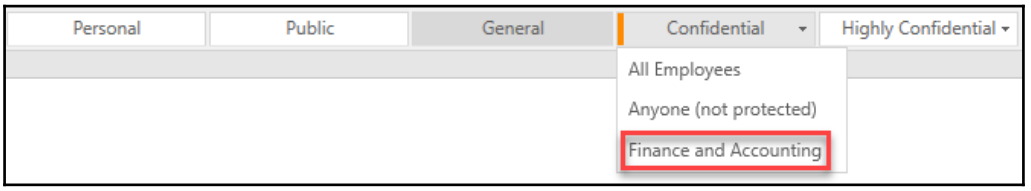
AIP is currently limited to providing only one sub-level label.

7. Leave the other settings as their defaults, as follows:



Scoping for Finance and Accounting users

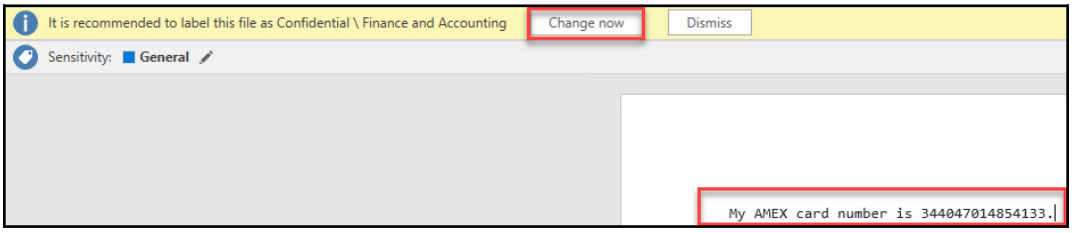
8. Log in as `Karim Manar` from the **Finance and Accounting** group on the second test client, and open Word.
9. You should see the new label, as follows:



Finance and Accounting label result

All other users will receive the default label set.

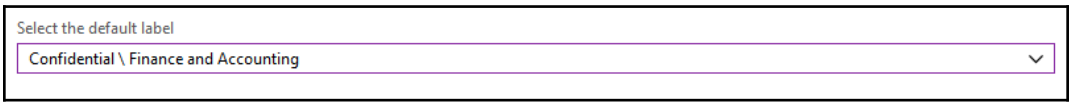
10. The next feature we already activated is the recommendation of the label if the specific condition matches.
11. Log in as `Karim Manar`.
12. Open Word and create a document with the content `My AMEX card number is 344047014854133`.
13. Save the document and the recommendation for the label appears as follows:



AIP label recommendation feature example

Remember, we start without encrypting the information to teach your users and to avoid problems with the end user experience and use of encrypted information. We also want to gather more information about the rules to have a clear identifier before starting automatic encryption.

With this option, you can also define a specific default label for each policy. This is a very helpful feature to support the usability of a specific department user, because the **Finance and Accounting** department handles mostly **Confidential** content. Define the default label, as follows:



Scoping the default label on custom policies

Create more sub-labels for the following departments to get an idea of how you can design your labels. In a real project, test and monitor the use of labels and leave it at a minimum, for example:

- **Strategy Consulting**
- **Senior Management**
- **Human Resources**
- **Executives**

If you are using more policies, the last policy wins in a conflicting setting.

The next screenshot shows the classification schema:

LABEL DISPLAY NAME	POLICY	MARKING	PROTECTION
Personal	Global		
Public	Global		
General	Global		
Confidential	Global		
Finance and Accounting	Finance and Accounting		
Strategy Consulting	Strategy Consulting		
Senior Management	Senior Management		
Human Resources	Human Resources		
Executives	Executives		
Recipients Only	Global	✓	
All Employees	Global	✓	
Anyone (not protected)	Global	✓	
Highly Confidential	Global		

Modified classification schema

You can use the same strategy for sensitive projects for other departments. Always think, discuss the real requirement for a new label, and assign it to the associated user group. It's very important to have an explicit usage and a proper handling for the user to provide a successful solution.

Using visual markings

The next option we can add is the marking option, to make the classification also visible in the content of the mail or document as a header, footer, or watermark. By default, the marking option is applied under the **Confidential** and **Highly Confidential** labels.

With the next steps in this section we will reach and view the following result:

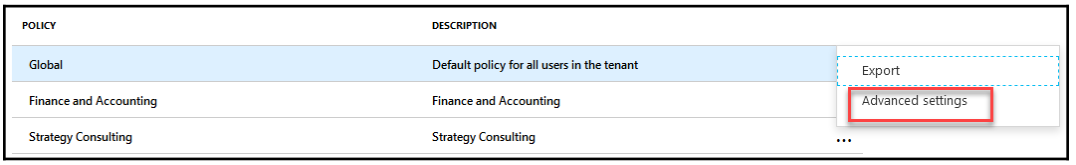
- View the default behavior for the **Confidential** label
- Open Word as Don Hall and apply the **Confidential | All Employees** label
- You will notice the footer contains **Classified as Confidential**

If you work with templates that already use the header/footer section, you need to test the templates to ensure that the classification doesn't crash your template design. By default, no watermark will be applied.

Use the following source to customize the visual markings: `https://bit.ly/2Cu8BnU`.

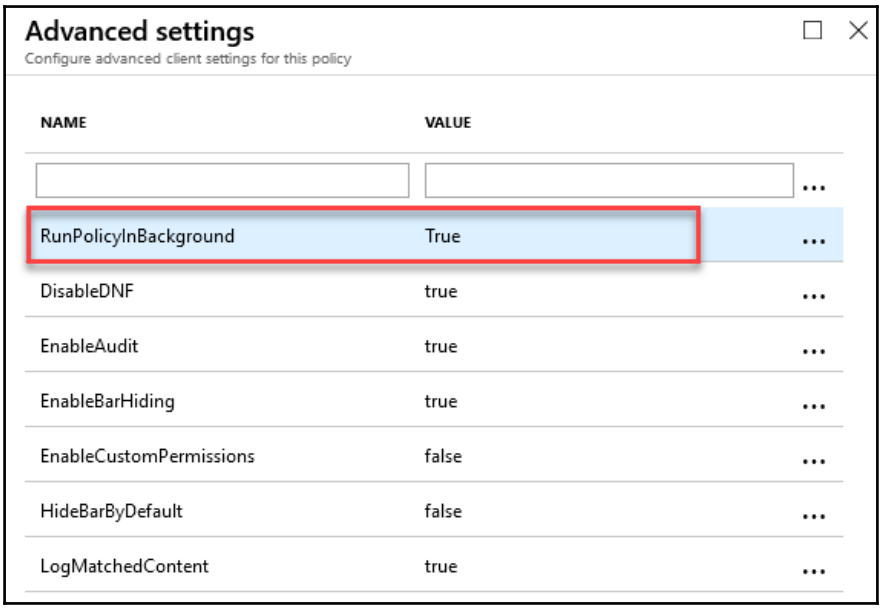
A typical example could be `Classified as ${Item.Label} from ${User.Name} at ${Event.DateTime}`.

If you are using the new **AutoSave** option, we recommend enabling the continuous classification feature in the **Advanced settings** with the following strings **NAME**: **RunPolicyInBackground**, **VALUE**: **True**, for example:



AIP policy advanced settings

You will find additional features in this area, as follows:



Configured advanced settings in the policy

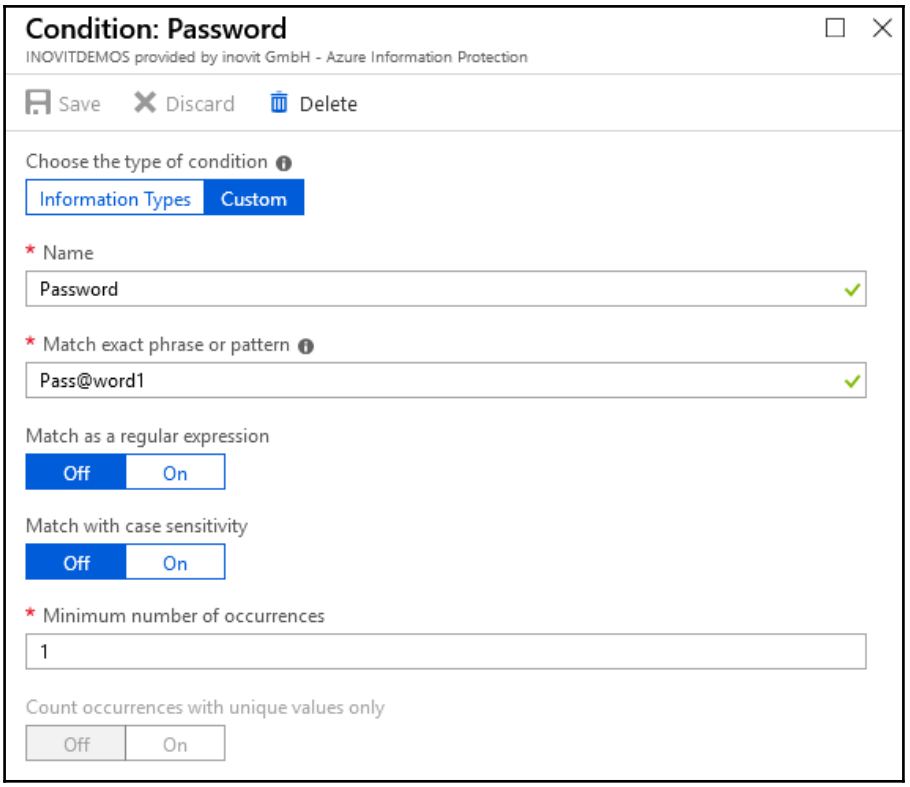
Next, we will explore the automatic classification feature.

Configuring automatic classification and protection

AIP also provides automatic classification. The feature requires the **Azure Information Protection P2** licensing. The feature is included in the **EMS E5** and **Microsoft 365 E5** plan.

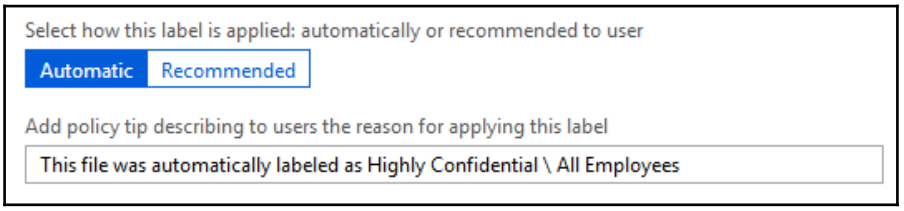
Start with clear rules and identifiers with automatic classification. For example, if a company uses an identifier in the document, such as `Approved for public consumption`, you can build an easy condition like we do in the following steps:

1. Edit the **Highly Confidential** | **All Employees** label and add the following condition:



Keyword definition for automatic classification

2. Change **Select how this label is applied: automatically or recommended to user** to **Automatic**, as follows:

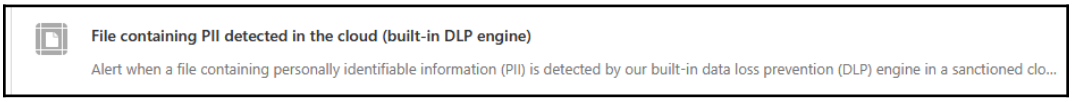


Activating automatic classification

3. Test the configuration with Don Hall.
4. Create a new Word document with `My password is Pass@word1` in the document.
5. The document will be classified automatically as **Highly Confidential | All Employees**.

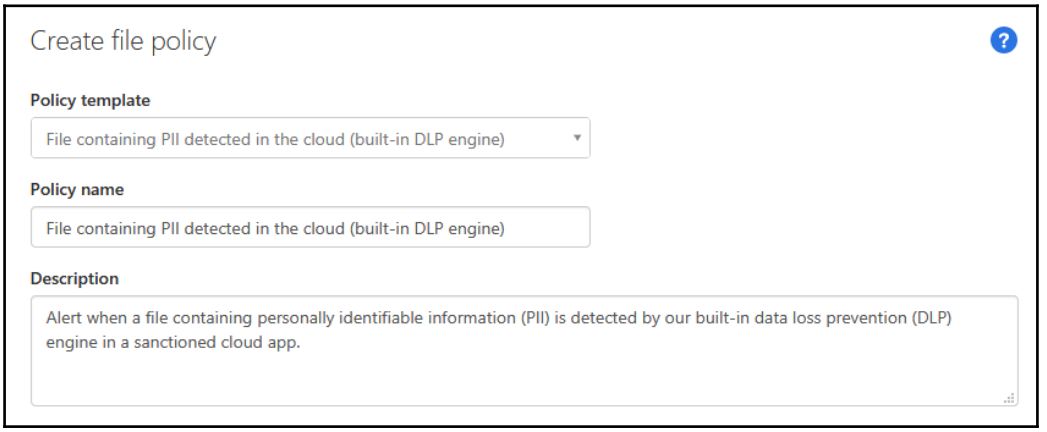
We can also use Cloud App Security to automatically classify and protect content by following these steps:

1. Open `https://portal.cloudappsecurity.com` as a global administrator.
2. Click **Create policies** in the **Get started** section.
3. Use the following template for our example (be sure that you put some Personal Identifiable Information PII files from the code package to your test user OneDrive):



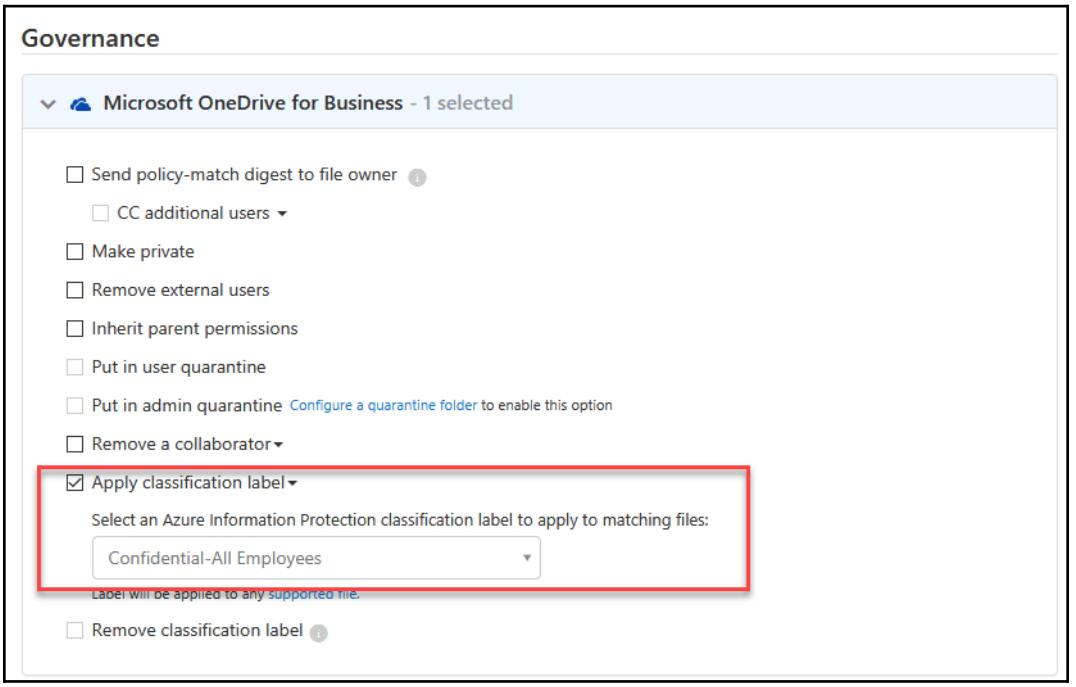
Cloud App Security PII scan result

4. Use the following settings in your new policy:



Policy creation in Cloud App Security to catch PII information

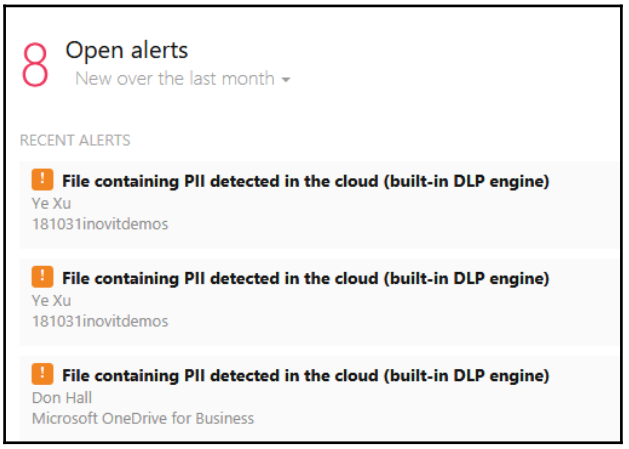
5. Choose **Microsoft OneDrive for Business** and choose the **Confidential-All Employees** label, as follows:



Cloud App Security automatic classification option

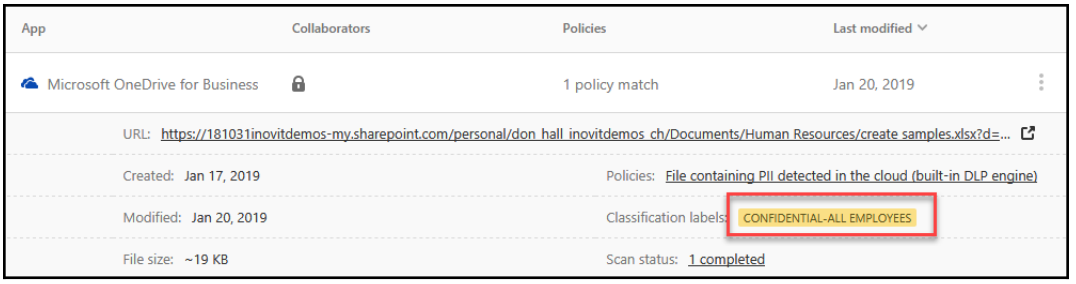
6. You will notice that if you click another app such as **Dropbox** or **Salesforce** the options are not available.

7. Next, move back to the dashboard and you will receive alerts with a match to our freshly created file policy, as follows:



Rule results after scanning

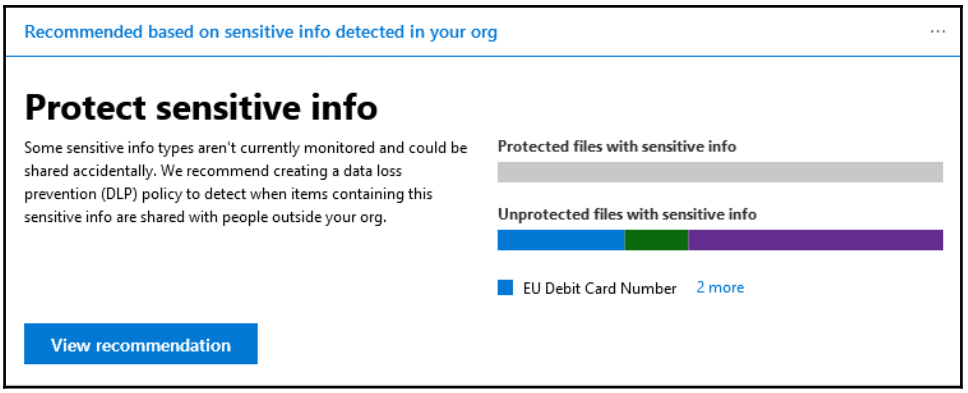
8. Click on the OneDrive alert and view the result; the file contains the PII information and it's classified:



Automatically classified document

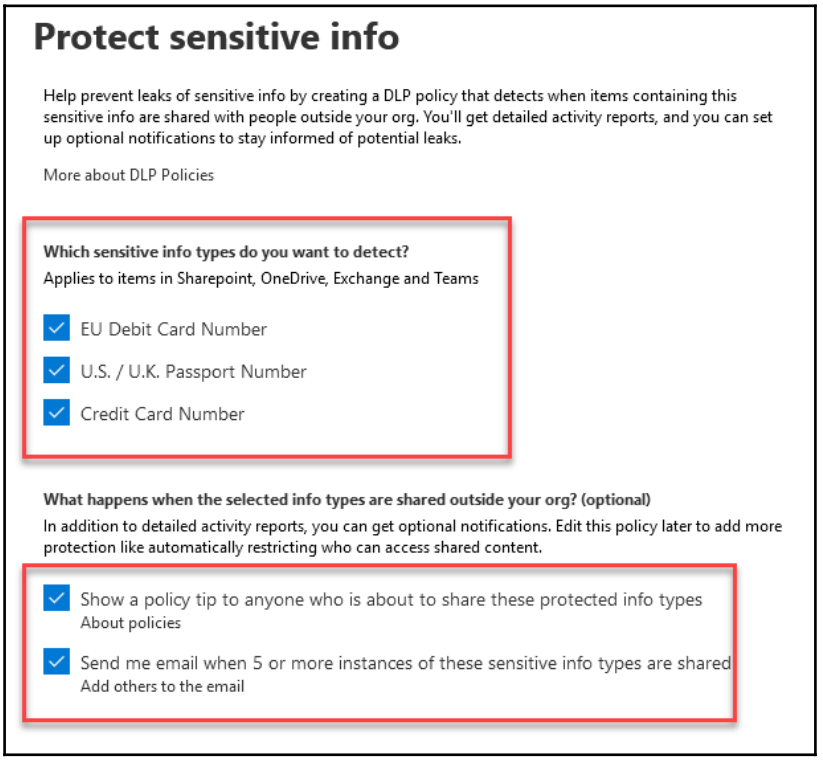
Another option to automatically classify and protect sensitive information can be found in Office 365 by using the following steps:

1. Navigate to the Office 365 portal at `https://portal.office.com` and sign in as a global administrator.
2. You will notice the following **Recommendation** to protect sensitive information. This is because the system has already detected some files you uploaded from the code package to OneDrive:



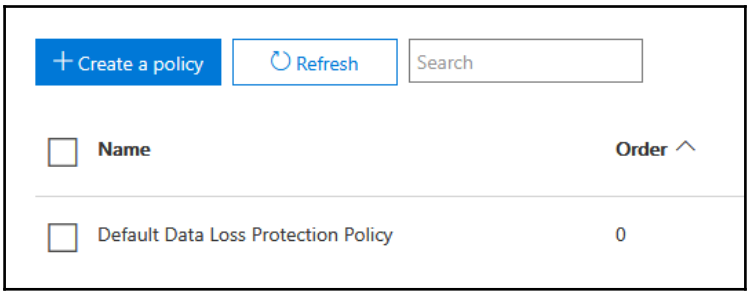
DLP options in Office 365

3. Click **View recommendation** and the system will give you the following option to create a policy to protect the information:



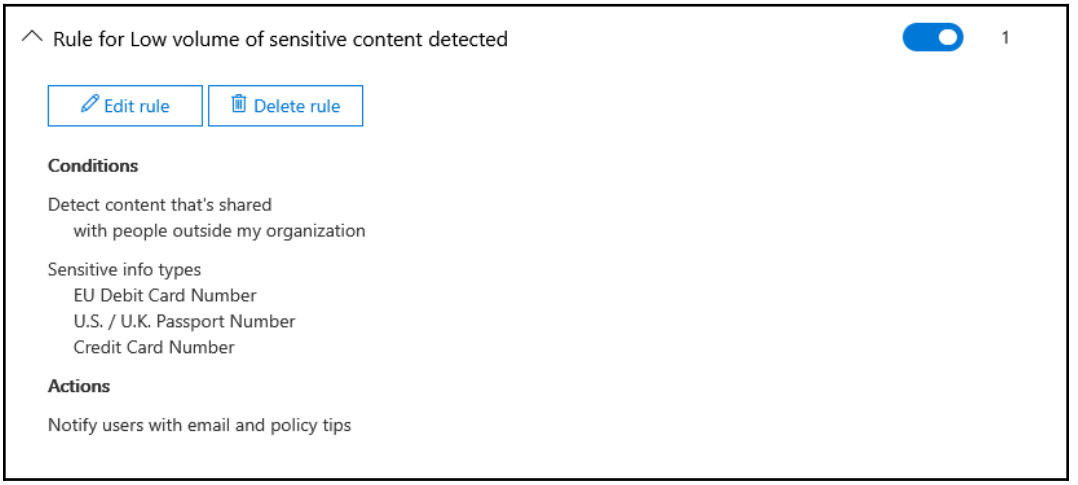
Scanning for sensitive information, such as credit card details

4. Click **Create a policy** at the bottom of the message.
5. You will find the newly created policy under `https://bit.ly/2FMYBJS`, and it will be called **Default Data Loss Protection Policy**:



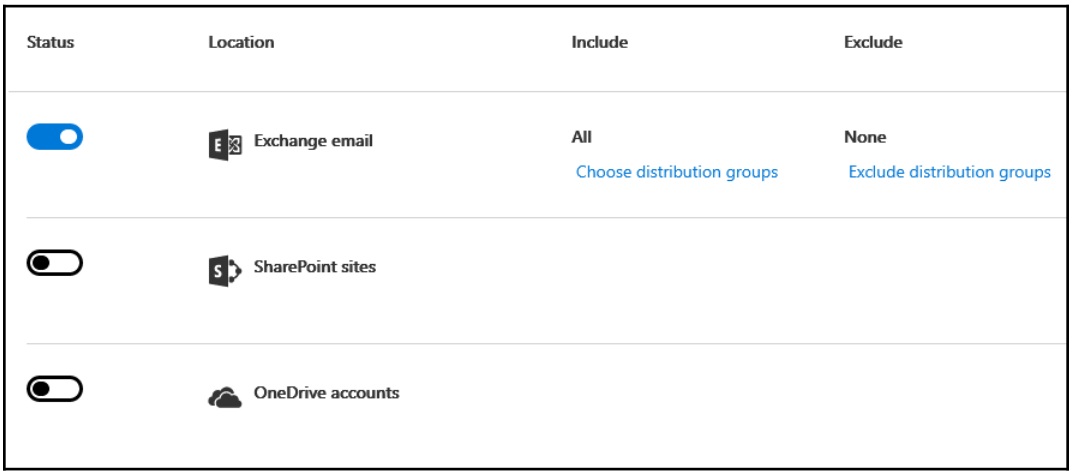
Policy creation wizard to use the rules

6. Edit the policy and view the settings, as follows:



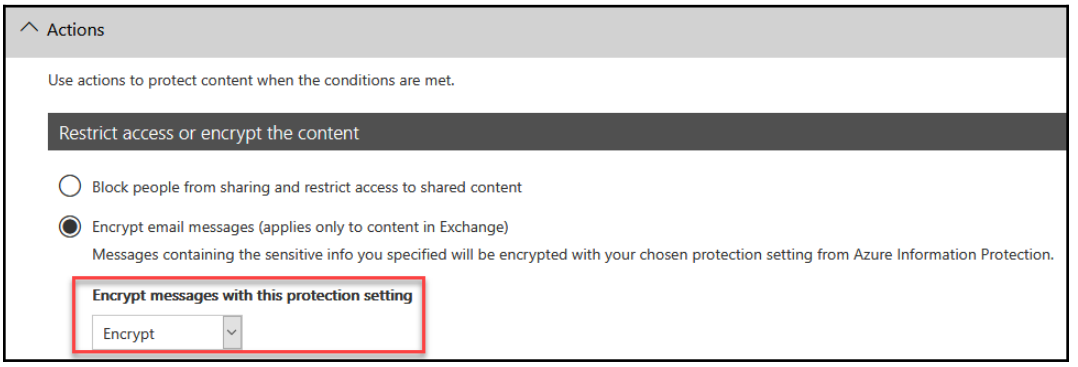
Policy summary

7. Edit the rule under **Location**; remove **SharePoint sites** and **OneDrive accounts**:



Location definition for the policy

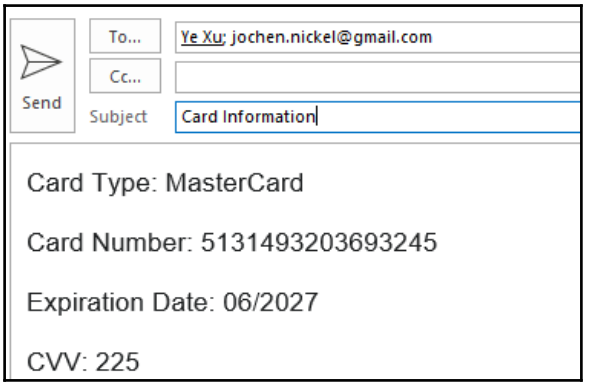
8. Edit the rule under **Actions**; choose **Encrypt** messages:



Enable encryption if sensitive content is detected

9. Let's test the policy with Don Hall on one of our test clients.
10. Open Outlook and send a message to Ye Xu and your private mail account with the following content.
11. You can create a credit card number on the following source (`https://bit.ly/2KeZHR1`):
    - **Card Type**: `MasterCard`
    - **Card Number**: `5131493203693245`
    - **Expiration Date**: `06/2027`
    - **CVV**: `225`

The following screenshot shows the mail with the sensitive content:

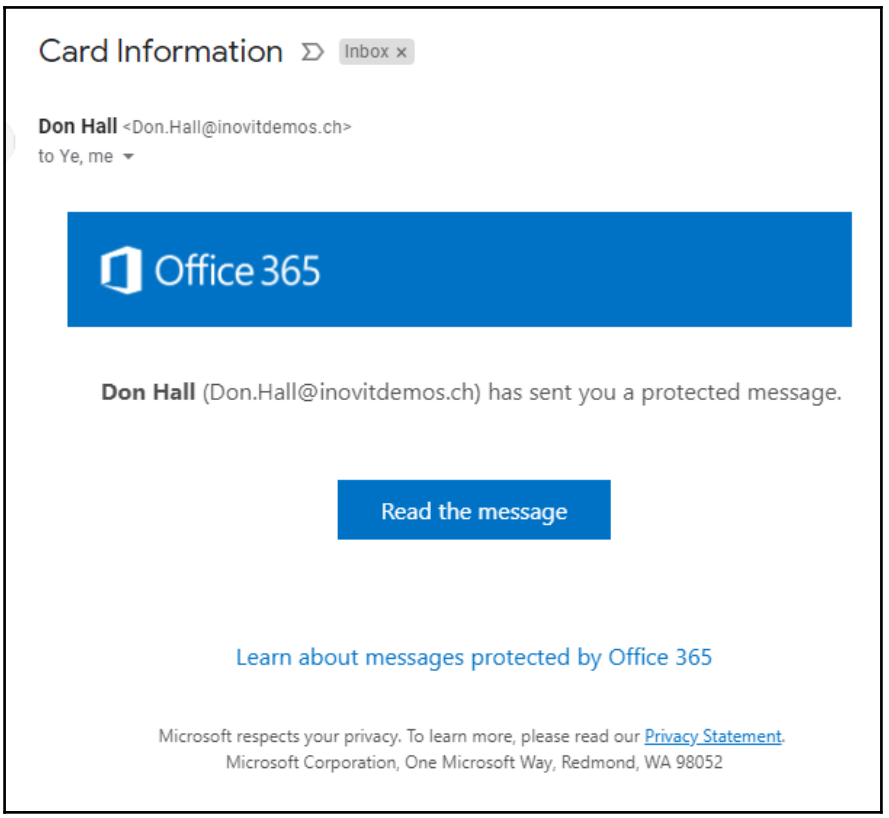


Test mail with sensitive content

12. Open your private mail account:

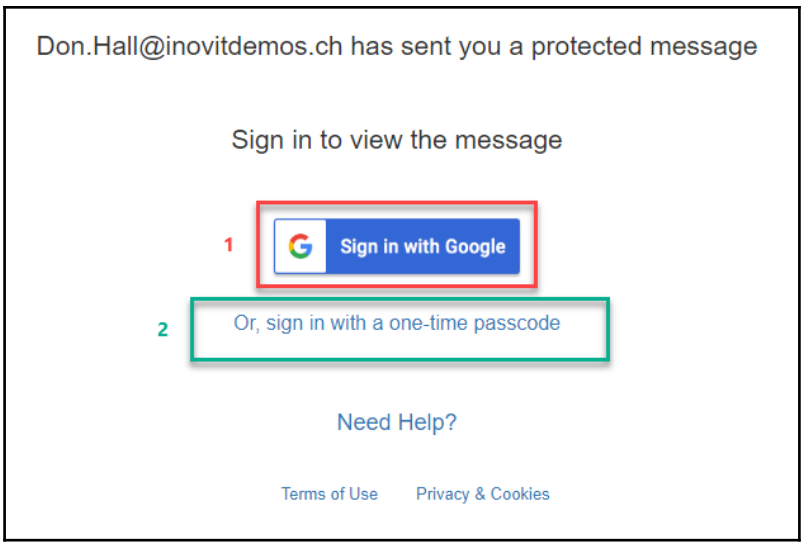
Private mail account expected result

13. Click **Read the message**:



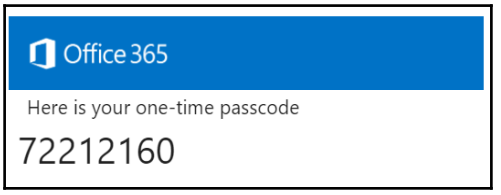
Read the encrypted message dialog

14. You will get two options to access the protected content:



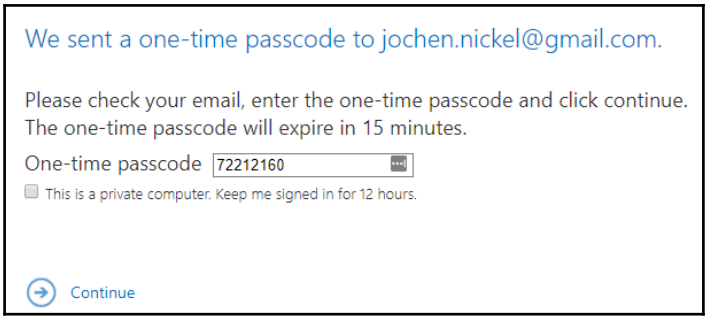
Sign in or one-time passcode authentication option

15. Choose the **one-time passcode** option to see the passcode behavior.
16. You will receive a mail with the passcode:



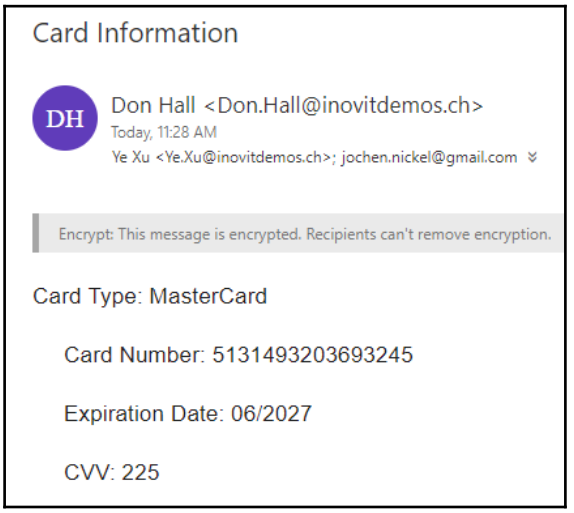
Received verification code

17. Enter the passcode and click **Continue**:



Code verification

18. And you can read the mail:

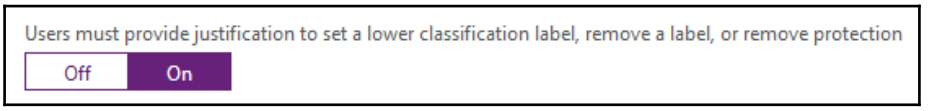


Message in plaintext

After working through a few automatic options, we will start using the justification option.

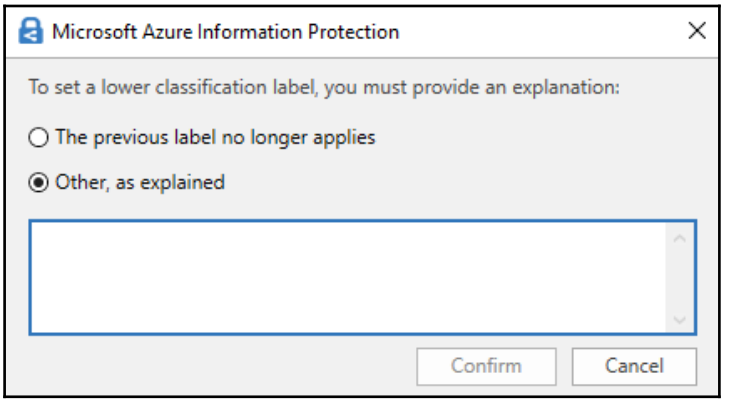
Using justification

The justification feature is a good option to change a recommended label or modify an existing label to address the sensitivity of a modified document. You can enable this feature under the policy settings:



Justification enabled

To test the functionality, open a document and downgrade it to the personal label:



Providing a justification

After a while, you will be able to see the following modification in the activity logs:

2019-01-19	10:53:16	don.hall@inovitdemos.ch	test.docx	Downgrade label	Public	No

Log information for the justification

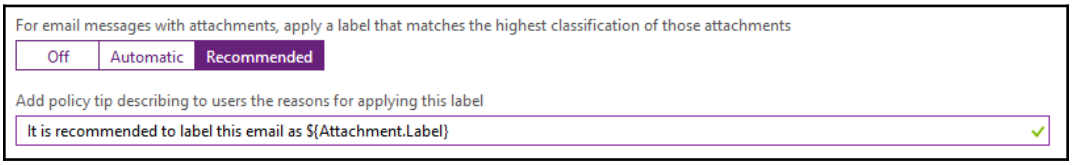
We recommend using the activity logs to provide information on your classification life cycle. You particularly need this information to improve your automatic rules. Maybe your conditions are no longer valid or not clear and data gets recommended for the wrong classification label.

Configuring protection options

Once your classification labels are appropriate and the users have been trained with the new methods, we can increase the protection options to achieve a more protected state.

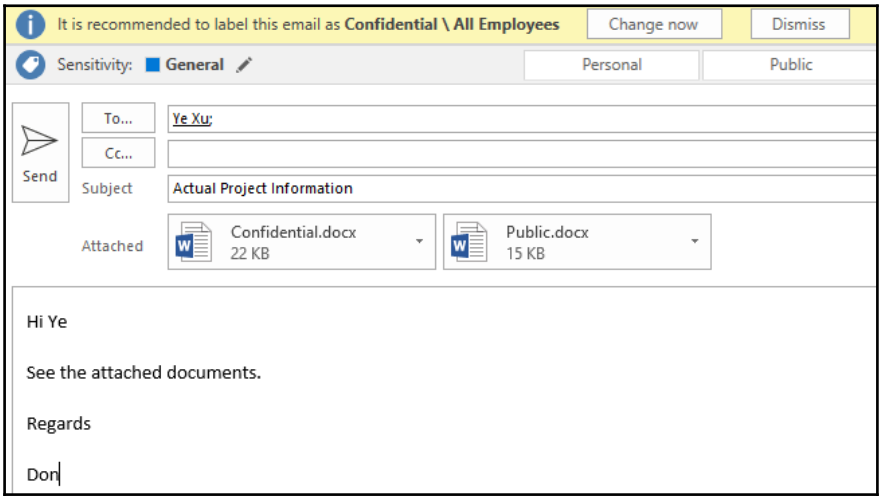
For email messages, we can use the option to increase the mail classification based on the highest attachment classification. We highly recommend using this as a recommended option and only if you have a crystal clear use case for the automatic option. With the following steps, you can configure this use case:

1. You can find the option in the policy configuration:



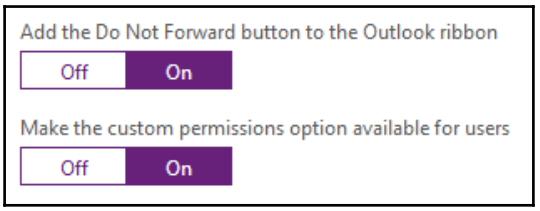
Recommended classification option

2. Log in as Don Hall to a test client and create two Word documents, one classified as `Public` and the other as `Confidential`.
3. Next, create a test email with the two attachments and send it to `Ye Xu`, as follows:



Recommendation in action

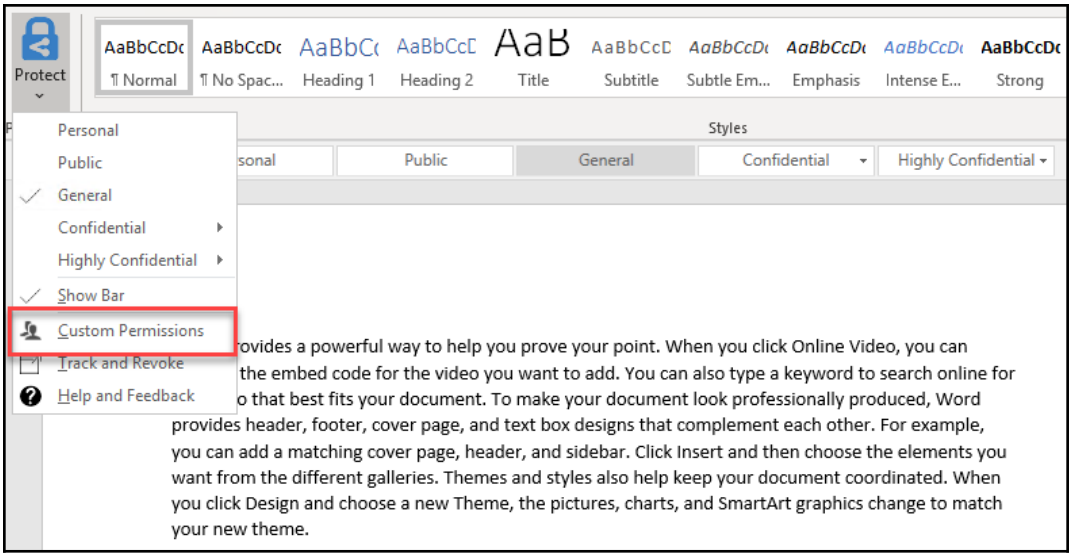
The following two options are useful for using custom protection:



Additional protection features for Office users

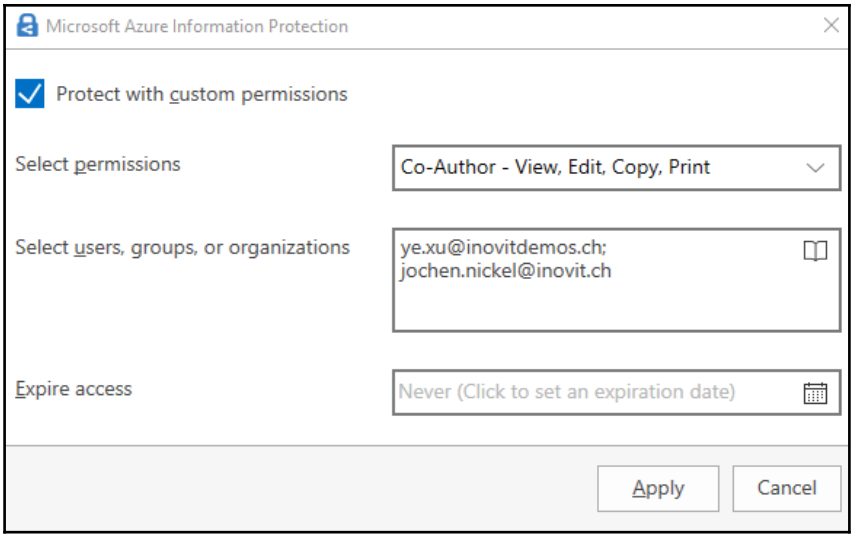
We will test the options with Don Hall in the following steps:

1. Log in as Don Hall and create a new Word document.
2. Click the **Protect** icon and use the **Custom Permissions**:



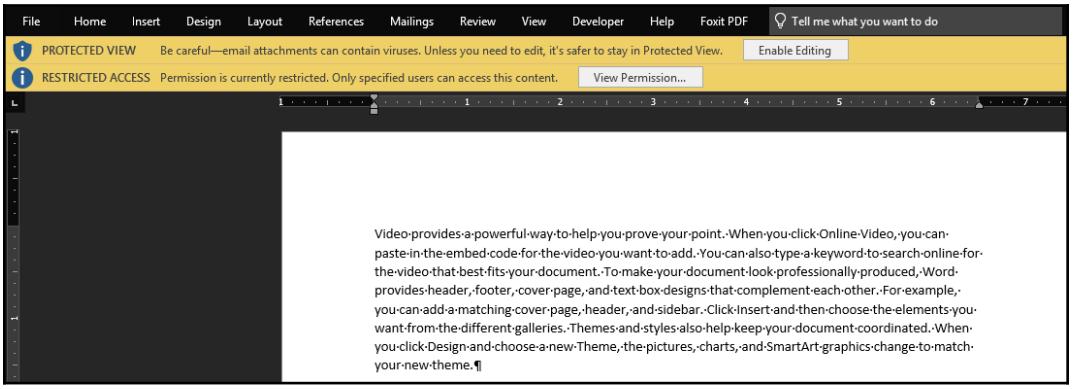
Custom permissions dialog

3. Assign co-author permissions to Ye Xu and one external user, as follows:



Using the custom permission options, including external accounts

4. Send the mail with the document attached to the two recipients.
5. Open the mail and the attachment on the external user's computer:

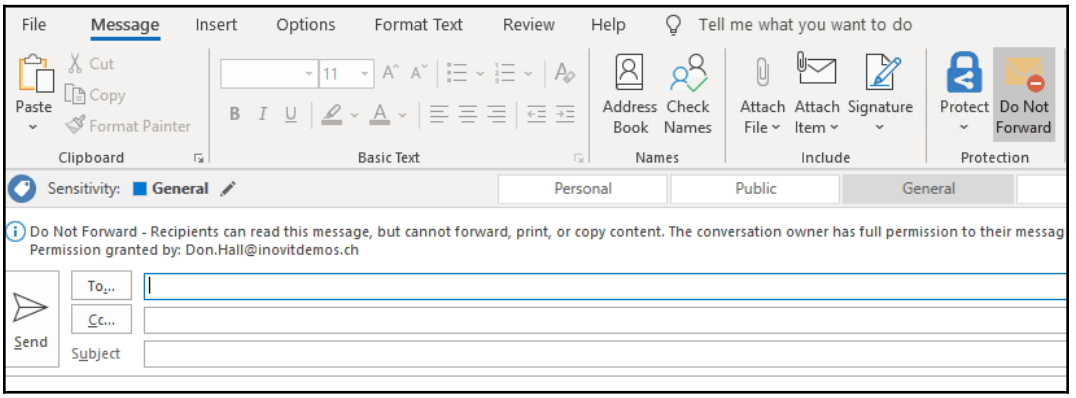


External users, view of the protected information

Be aware that the behavior of Azure RMS protected content is slightly different and depends on the configuration and RMS enablement of the recipient. If the other user also uses Office 365, there should be a transparent usage for the recipient without login prompt. When the user is not using any Microsoft cloud service and no Azure AD, the recipient will be automatically processed to create an ad hoc Azure AD account in order to use the Azure RMS functionality. Prepare your external recipients and include them in the training you do for your internal users:

If you want to regulate all Azure RMS protected content, you should build a label for the process and hide the button.

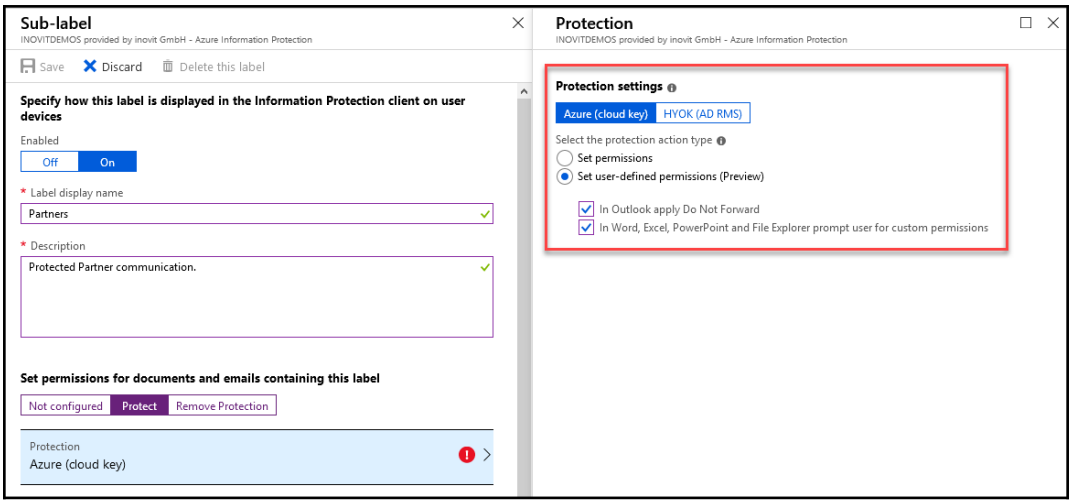
1. Use the **Do Not Forward** button in Outlook to encrypt the message:



Do Not Forward button in Outlook (not in other Office products)

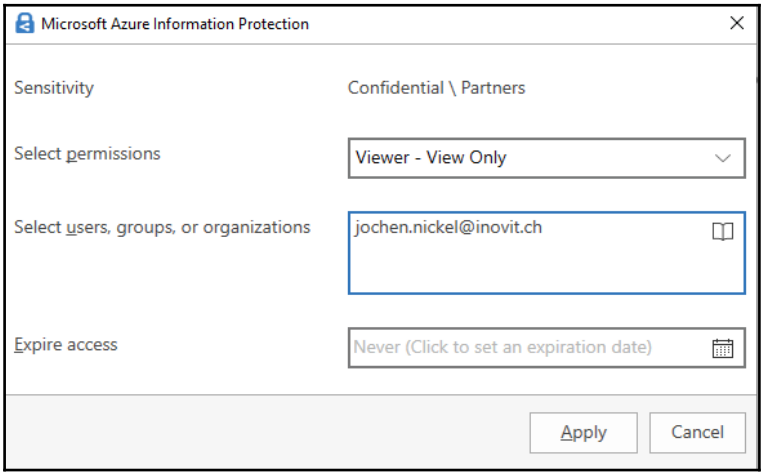
Provide a fully explained Business to Business and Business to Customer protection scenario.

2. You can provide this functionality inside your labels.
3. We can create a custom label called `Partners` under **Confidential** and assign it to the global policy, as follows:



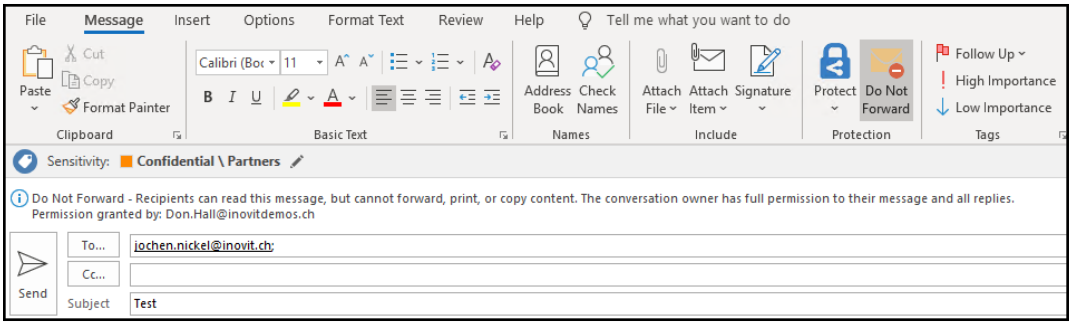
Using the custom protection options in a label

4. Use a custom footer with the following text "classified as partner confidential".
5. Log in as Don Hall to a test client.
6. Create a Word document and test the new functionality, as follows:



Testing the custom permission option in Word

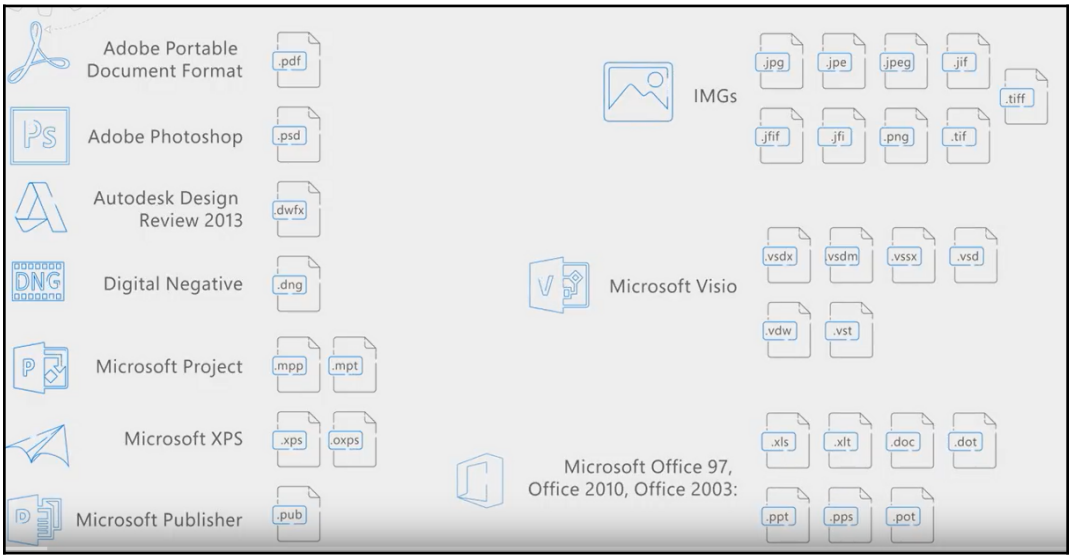
7. Create a new mail in Outlook and test the new functionality, as follows:



Testing the Do Not Forward button in Outlook

Keep in mind that online Office components can't use protected information.

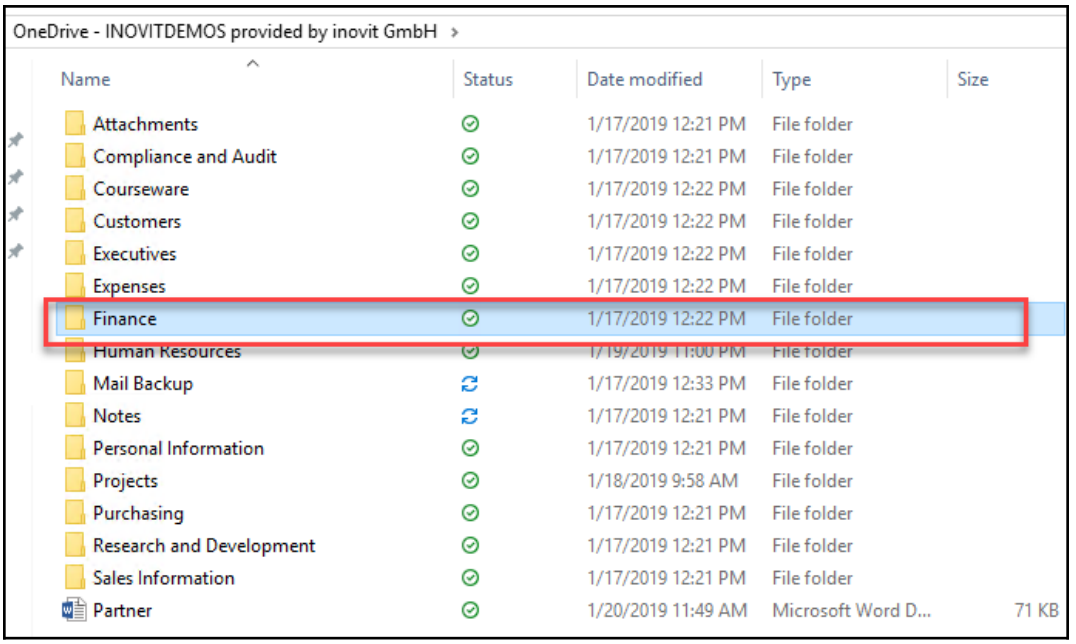
Actually, Microsoft supports many different applications and file formats to classify and protect; you can see the applications in the following screenshot:



Document types supported by Azure RMS

Microsoft also provides a universal solution over the integration that we can use to classify and protect sensitive data. The following steps show the procedure:

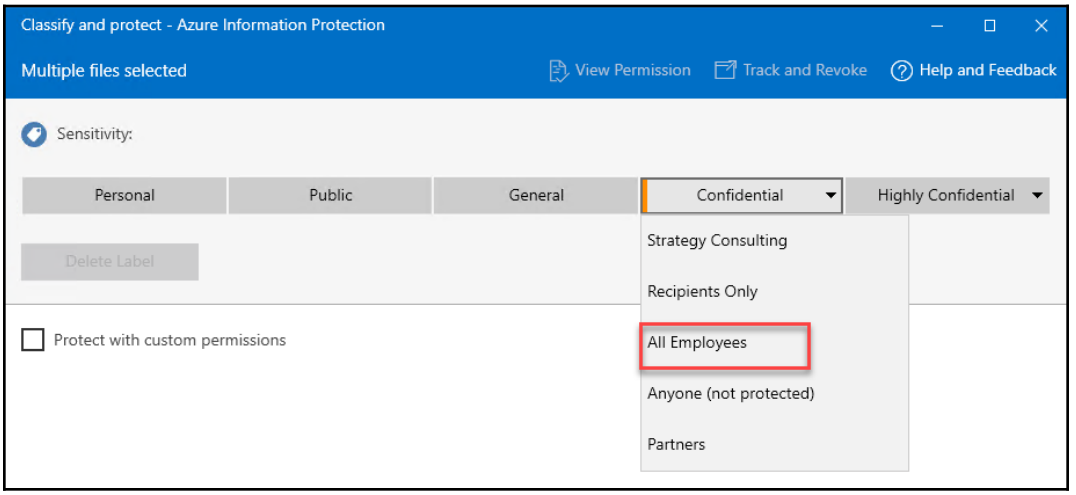
1. Log on as Don Hall and open your `OneDrive` folder, as follows:



Using the desktop classification option

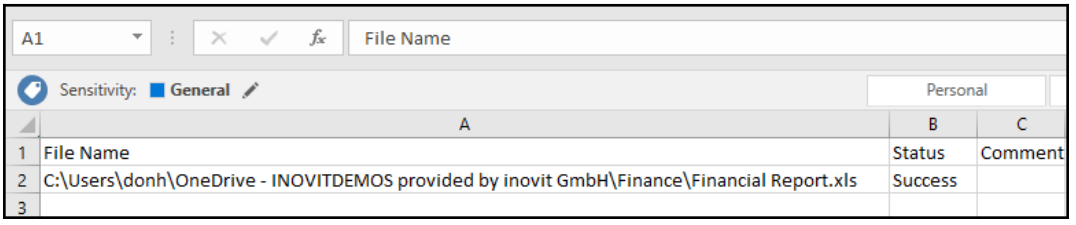
2. Right-click on the `Finance` folder and choose **Classify and protect**.

3. Use the **Confidential | All Employees** label to basically classify the documents, as follows:



Choosing the All Employees label to classify the information

4. Later, if you have your complete process in place, you can provide a specific label and protect the information.
5. You will be able to see a report of all actions:



Specific label reporting

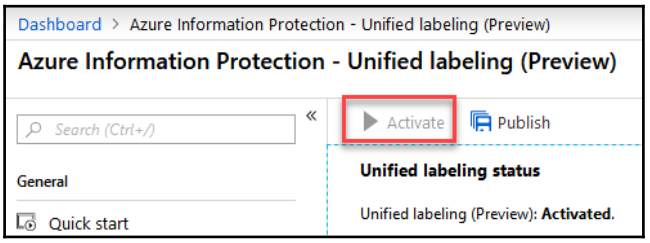
With this option, you can classify and protect several different files, based on the automatic rules or your manually chosen label.

Activating unified labeling

With the strategy of Microsoft to build a complete Information Protection solution, Office 365 and AIP labels will be merged together. With this option, you will have full integration with the most important services.

In the following steps, we will activate the unified labeling functionality:

1. You can activate unified labeling by opening `https://bit.ly/2RHzZZZ` with global administrator credentials.
2. Click the **Activate** button and **Publish** your labels, as follows:



Unified labeling activation

3. Navigate to `https://protection.office.com/` and click on **Classifications and Labels** to review the migrated labels.

Next, we will let you run through a lab challenge to discover more AIP capabilities.

Lab challenge

Build up a complete classification schema with several processes using the actually created groups and test all the different functionalities. Try to implement the following processes:

- Partner and customer processes
- Sensitive project content management
- Internal collaboration processes between departments
- Email protection for external users
- Protecting your SharePoint/OneDrive sharing of information/data
- Prevent data leakage in other platforms, such as Dropbox and Salesforce

Good luck!

Summary

Working through this chapter enabled you to start to optimize your Information Protection solutions inside your organization or with your customers. We provided a starting lab environment to test all the required functionality and processes of AIP. With the key PowerShell cmdlets, you should be up and running, and ready to start your configuration tasks. Furthermore, with the overview of typical configuration tasks, we provided you a few examples of the experiences we learned during our many projects.

In the last chapter of the book, we will work through an AIP-enabled example application to give you more insights into the AIP technology.

16
Azure Information Protection Development

This chapter is a good starting point for anyone who wants to dive deeper into the Azure Information Protection technology to get more information for troubleshooting or supporting a solution. This chapter will also help you write an application to help your customers to fulfill your organization's needs. Using the development resources of Azure Information Protection will give you more in-depth knowledge about this technology.

In this chapter, we'll provide you with an overview of the different development resources to help you start your journey into the different Azure Information Protection development options. We'll offer you the starters to prepare your development environment and give you some examples so you can start analyzing this excellent service. You'll get information about the Microsoft Information Protection SDK, the use of PowerShell, and the other SDKs that are available to help you begin your journey.

This chapter is organized into the following sections:

- Microsoft Information Protection solutions
- Understanding the Microsoft Information Protection SDK
- Preparing your Azure AD environment for tests
- Using MIP binaries to explore functionality
- Using PowerShell with Azure Information Protection
- Overview of the RMS 2.1 and 4.2 SDKs

Lets look at the technical requirements you need to start developing on our lab environment.

Technical requirements

In this chapter, you can use the **YD1APP01** server, on which Visual Studio 2017 is already installed in `Chapter 7`, *Deploying Solutions on Azure AD and ADFS*. Additionally, you need to install the Information Protection SDK 2.1 from `https://bit.ly/298QzMn` to run any development. Also, download the examples from `https://bit.ly/2CBVvoE`, `https://bit.ly/2RHCjjO`, and `https://bit.ly/2HtYKEj` to the server in your Visual Studio project directory, such as `C:\Users\cloudadmin.INOVITDEMOS\Documents\Visual Studio 2017\Projects>`, which you can use to experiment with the framework.

In the next section, we'll dive into the Microsoft Information Protection solutions to get the knowledge about all the related technologies.

Microsoft Information Protection solutions

Azure Information Protection itself is built to classify, label, and protect files in Office 365 and many other applications. The following list shows the different solution components and their relation to Azure Information Protection:

- **Microsoft Cloud App Security**: For protecting cloud applications with direct integration of Azure Information Protection.
- **Conditional Access**: To control access to sensitive information where no direct integration into Azure Information Protection is required.
- **SharePoint**: To deliver RMS-protected libraries and groups to provide access control.
- **Office 365 Message encryption**: To send encrypted emails inside or outside your organization with a direct integration into Azure Information Protection.
- **Office 365 Data Loss Prevention**: Is built to prevent data loss over Exchange and SharePoint Online, including OneDrive for Business with a direct integration into Azure Information Protection.
- **Office 365 Advanced Data Governance**: Provides the retention and deletion policies for sensitive information with a direct integration into Azure Information Protection.

- **Azure Security Center Information Protection**: Assigns classification of structured data, such as Azure SQL or SQL servers, including other Azure repositories with no direct integration into Azure Information Protection.
- **Windows Information Protection**: Is built to separate the private and business context on a mobile phone or PC.
- **Office Apps**: Used to protect sensitive information in Excel, PowerPoint, Word, and Outlook with an indirect integration into Azure Information Protection.
- **Adobe PDFs**: Azure Information Protection delivers the native support to label and protect Adobe PDF's.

Overall, Microsoft provides Microsoft SDKs to help developers to work on solutions for traditional on-premises infrastructure, cloud platforms, and modern applications on mobile devices. Lets understand more about it.

Understanding the Microsoft Information Protection SDK

The Microsoft Information Protection SDK extends labeling and protection functionality so you can provide a consistent experience in cross-platform scenarios, and it provides a comprehensive set of capabilities. With the SDK, you can extend the classification, labeling, and protection to any other app and service. In general, Microsoft Information Protection solutions are compatible with traditional Active Directory RMS infrastructures. You experienced this with the **Hold Your Own Key** (**HYOK**) functionality.

The Microsoft Information Protection SDK is available on the following:

- macOS, Linux, and Windows
- A preview for Android, IOS, and other platforms

The SDK supports user and service applications, including support for multi-tenancy. User applications include common authoring tools, such as Office and Adobe. Labeling structured data is also supported. Service applications mostly run in the background, such as DLP, **cloud access security brokers** (**CASBs**), and eDiscovery features. These applications authenticate using OAuth.

In detail, the APIs provide the following capabilities:

API	Functionality	Use cases
Protection	• Accepts plaintext content and returns a publishing license • Allows ciphertext and publishing licenses • Returns plaintext • Rights enforcement if needed	• Encryption of any data • No key exchange management required • For example, security software that needs to inspect data in transit
Policy	• Computes actions that policy should take • Submission enablement for events in the audit pipeline • Application of label metadata if needed	• CAD and CAM applications for classifying parts or components • DLP platforms for reading labels and monitoring your services or networks
File	• Support for well-known file types • Allows read, write, or delete labels • Enables protection removal • Applies metadata provided by the API	• Word, Excel, PowerPoint, or others • Scanners that need to label files

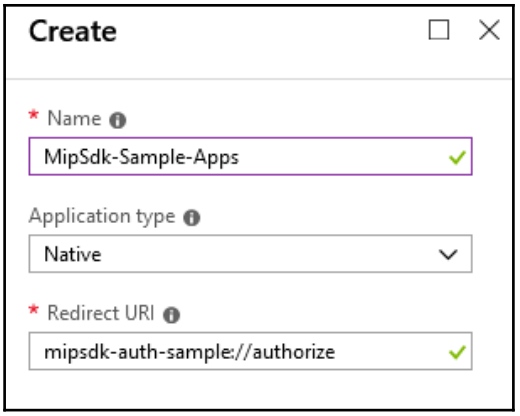
In the next section, we'll provide a sample app, `MipSdk-FileApi-Cpp-Sample-Basic`, from Microsoft and help you to start using the SDK and explore the different functionalities.

Preparing your Azure AD environment for tests

In this section, we'll adjust our Azure AD environment to run code from the Microsoft Information Protection SDK against our Azure Information Protection infrastructure. As usual, it starts with the creation of an Azure AD app:

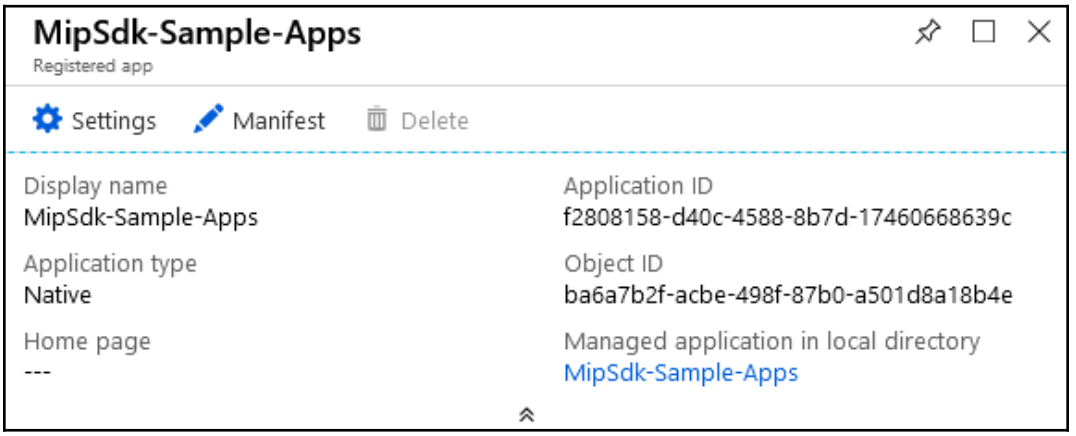
1. Log in with global administrator credentials to the Azure portal at `https://portal.azure.com`.
2. Navigate to the Azure AD blade.
3. Click **App registrations** to create a new application.
4. Click **New Applications registration**.

5. Use the following settings:



Example app properties

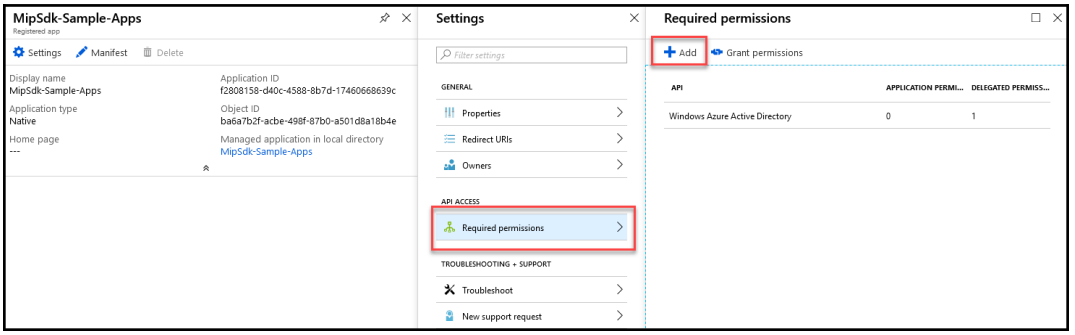
6. Click the **Settings** button on the registered application:



App Settings option

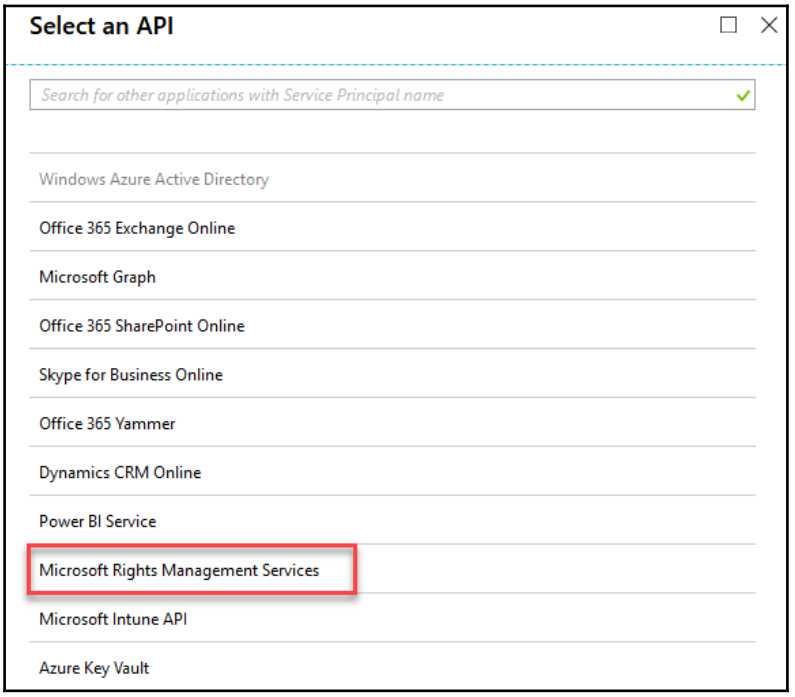
7. Click the **Required permissions** section for API access.

8. Click **Add**:



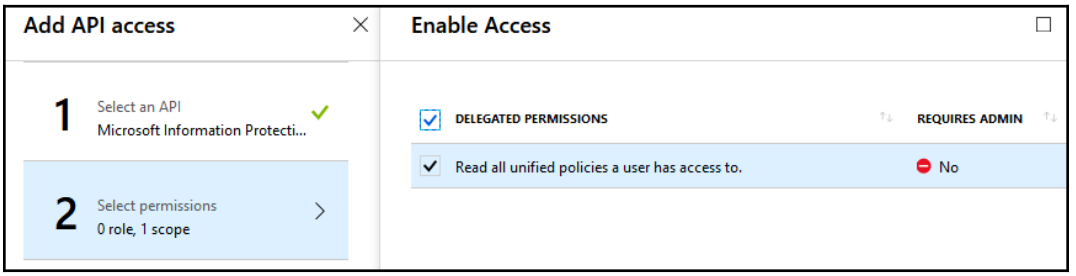
Required permissions configuration

9. Click **Select an API**. If needed, use the search field to find **Microsoft Rights Management Services**.
10. Select the **Microsoft Rights Management Services** API:



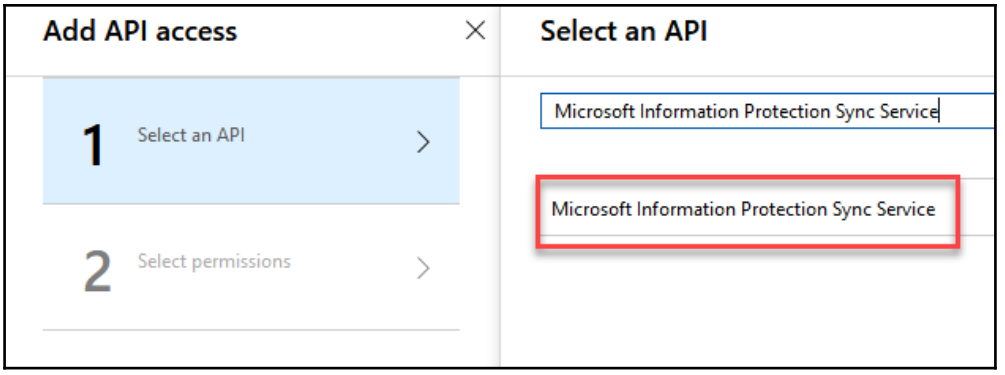
Choosing the Microsoft RMS API

11. Under the **Select permissions** section, use the **Create and access protected content for users** permission under the **DELEGATED PERMISSIONS** options.
12. Click **Select** and **Done**:



Permission scoping

13. Click the **Add** button.
14. Click the **Select an API** option to choose the correct API.
15. Search the **Microsoft Information Protection Sync Service** and select the service:



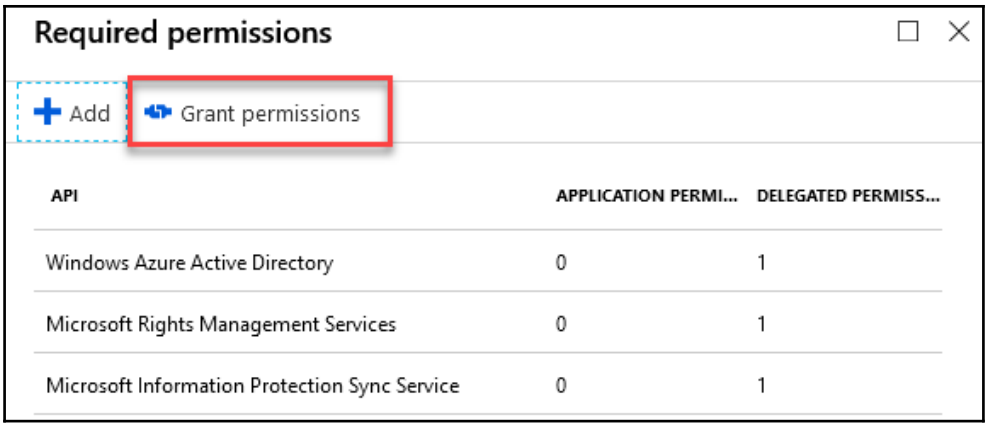
Select the Microsoft Information Protection Sync Service API

16. Under the **Select permissions** option, select **Read all unified policies a user has access to.** under **DELEGATED PERMISSIONS**.

17. Click **Select** and then press **Done**:

Delegated permission configuration

18. In the **Required permissions** blade, click the **Grant permissions** button and confirm the dialog:



Permission granting procedure

19. Copy the application ID in line `57` of the `main.cpp` file, and your test user into line `64`, to configure the application:

```
54    // Create the mip::ApplicationInfo object.
55    // Client ID should be the client ID registered in Azure AD for your custom application.
56    // Friendly Name should be the name of the application as it should appear in reports.
57    mip::ApplicationInfo appInfo{"f2808158-d40c-4588-8b7d-17460668639c", "MipSdkFileApiCppSampleBasic" };
58
59    // All actions for this tutorial project are implemented in samples::file::Action
60    // Source files are Action.h/cpp.
61    // "File" was chosen because this example is specifically for the MIP SDK File API.
62    // Action's constructor takes in the mip::ApplicationInfo object and uses the client ID for auth.
63    // Username and password are required in this sample as the oauth2 token is obtained via Python script and basic auth.
64    Action action = Action(appInfo, "don.hall@inovitdemos.ch","K        3");
```

Modification of the example app

Now that we have prepared our Azure AD environment, you can start exploring the code samples.

Open the `MipSdk-FileApi-Cpp-Sample-Basic.vcxproj` file with Visual Studio and start with the following steps:

1. Copy the label ID to the clipboard or Notepad.
2. Paste the label into the input prompt.
3. The application asks for the path to a file. Enter the path to your specific Office document.
4. The application will display the name of the currently applied label.
5. Open the file in a viewer that supports the labeling or protection feature.

Next, we will explore the functionality of the Microsoft Information Protection (MIP) binaries to see the broad functionality that you can include in the development of your solution.

Using MIP binaries to explore functionality

With the Microsoft Information protection binary from the SDK, we can explore the new functionalities of Microsoft's unified Information Protection solution. The binaries deliver all the examples needed to start using the different functions like gathering the actual status of a file, classifying and protecting a file, and bulk decrypting files. In the following section, we will focus on the file_example.exe binary which delivers all this basic functionality. This example will give you ideas for developing own applications that use the SDK and the Information protection functions.

We downloaded the MIP binaries to explore features. You can use the following steps to get more insight into features:

1. Open PowerShell and navigate to the sample files.
2. Execute the `.\file_sample.exe` binary to view the functionality you can test:

```
PS C:\Python27> .\file_sample.exe
Microsoft Information Protection File SDK Sample Version: 1.0.54
Usage:
  file_sample [OPTION...] <Extra args>

  -f, --file File path         Path to the file to work on.
  -g, --getfilestatus          Show the labels and protection that applies on
                               the file.
  -s, --setlabel arg           Set a label with <labelId>. If downgrading
                               label - will apply <justification message>, if
                               needed and specified.
  -d, --delete                 Delete the current label from the file with
                               <justification message>, if needed and specified.
  -p, --protect arg            Protect with custom permissions protection to
                               comma-separated user list.<rights> as
                               permissions to those users
  -r, --rights arg             Comma-separated list of rights to users
      --templateid arg         Protect using Template ID
  -l, --listlabels             Show all available labels with their ID
                               values.
  -u, --unprotect              Remove protection from the given file.
      --standard               The label will be standard label adn will
                               override standard label only.
      --privileged             The label will be privileged label and will
                               override any label.
      --auto                   The label will be standard label and will
```

Application command-line options

With this binary, you can start with the following commands:

- Get a list of labels:

```
./file_sample --username jenny.green@leano.ch --password
"YOURPASSWORD " --file "PATH" --listlabels
```

- Set a label:

```
./file_sample --username jenny.green@leano.ch --password
"YOURPASSWORD" --file "PATH" --setlabel <GUID>
```

- Read a label:

```
./file_sample --username jenny.green@leano.ch --password
"YOURPASSWORD" --file "PATH" --getfilestatus
```

After working with the MIP file_sample.exe example, we will introduce you the several capabilities of PowerShell to administer and develop your own solution. The functionality is similar to the binaries, but maybe easier to understand for an administrator or PowerShell experienced person.

Using PowerShell with Azure Information Protection

PowerShell provides you managing capabilities for Azure Information Protection. In special, to handle custom classification and protection solutions you need to be able to use PowerShell to solve your challenges like, labeling and protecting files on a file share or single computer. With the following cmdlets you have the basic toolset to do the most administrative tasks with PowerShell. We used the cmdlets for example for providing a monitoring solution of a single folder and labeling and encrypting the files in the folder based on keywords. With the following commands, you can start to explore more about Azure Information Protection's capabilities and technology:

- `Import-Module AzureInformationProtection`
- `Get-AIPFileStatus`: Used to identify all files with a specific label
- `Set-AIPFileClassification`: Used to inspect file contents and automatically label unlabeled files
- `Set-AIPFileLabel`: Used to apply a specified label
- `Set-AIPAuthentication`: Used to label files non-interactively (scripting/scheduler)

Next, we will see some useful PowerShell cmdlets to administrator Azure RMS.

Useful Azure RMS cmdlets

In the following section, we will provide a service principal that is needed for using the scripts to administrate Azure RMS. The following commands are needed to create a new service principal:

```
$ServicePrincipalName="RMSPowerShell"
Connect-AadrmService
$bposTenantID=(Get-AadrmConfiguration).BPOSId
Disconnect-AadrmService
Connect-MsolService
New-MsolServicePrincipal -DisplayName $ServicePrincipalName
```

The following output is expected:

```
PS C:\> $ServicePrincipalName="RMSPowerShell"
PS C:\> Connect-AadrmService
A connection to the Microsoft Azure AD Rights Management (AADRM) service was opened.
PS C:\> $bposTenantID=(Get-AadrmConfiguration).BPOSId
PS C:\> Disconnect-AadrmService
Connection to the AADRM service closed.
PS C:\> Connect-MsolService
PS C:\> New-MsolServicePrincipal -DisplayName $ServicePrincipalName
The following symmetric key was created as one was not supplied e[redacted]k=


DisplayName           : RMSPowerShell
ServicePrincipalNames : {701[redacted]c936b98d9df4}
ObjectId              : e5df[redacted]a2bbf7041ef
AppPrincipalId        : 7017[redacted]936b98d9df4
TrustedForDelegation  : False
AccountEnabled        : True
Addresses             : {}
KeyType               : Symmetric
KeyId                 : 4c867324-412c-41f5-91c6-e62a8458a72d
StartDate             : 1/22/2019 10:31:47 PM
EndDate               : 1/22/2020 10:31:47 PM
Usage                 : Verify
```

Creation result of the new service principal

Copy the generated symmetric key value:

```
$symmetricKey="<value from the display of the New-MsolServicePrincipal
command>"
 $appPrincipalID=(Get-MsolServicePrincipal | Where { $_.DisplayName -
eq $ServicePrincipalName }).AppPrincipalId
 Set-RMSServerAuthentication -Key $symmetricKey -AppPrincipalId
$appPrincipalID -BposTenantId $bposTenantID
```

The following output is expected:

```
$symmetricKey="eGr+Q                              iMF2Mrj+vk="
$appPrincipalID=(Get-MsolServicePrincipal | Where { $_.DisplayName -eq $ServicePrincipalName }).AppPrincipalId
Set-RMSServerAuthentication -Key $symmetricKey -AppPrincipalId $appPrincipalID -BposTenantId $bposTenantID
```

Key information

Now, you can start doing the following commands using the service principal account:

- Enumerate all `RMSTemplate`:

```
Get-RMSTemplate
```

- Protect a file:

```
Protect-RMSFile -File C:\Test.docx -InPlace -TemplateId
<yourtemplateid>
```

- Get the status of a file:

```
Get-RMSFileStatus -File C:\Test1.docx
```

- Unprotect a file:

```
Unprotect-RMSFile C:\test.docx -InPlace
```

Now that we've started using PowerShell to work with Azure Information Protection and Azure Rights Management Service, we'll give you some useful information about the other SDKs that are available to develop against the services and increase your knowledge.

Overview of the RMS 2.1 and 4.2 SDKs

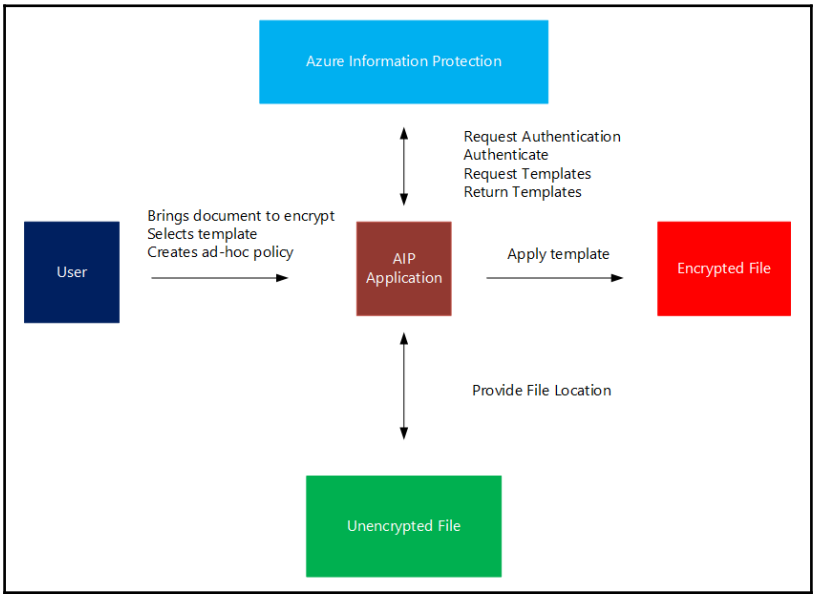
Two active generations of RMS SDKs are available to developers and used for the following developments:

- Microsoft Rights Management SDK 4.2 for Android, iOS/macOS, Windows devices, and Linux
- Microsoft Rights Management SDK 2.1 for Windows Desktop Client
- AD RMS SDK is superseded

The following improvements were included with the 4.2 version:

- Hybrid support AD RMS and Azure RMS (AD RMS' mobile device extension is needed to make AD RMS available on mobile devices and provides the required authentication methods)
- Access protected content offline
- Bring your authentication library
- Bring your user interface
- Redesigned API

You can use the following example design to develop your application:



Example of design for an application with the feature specification

The example above demonstrates how a user can encrypt a document. For this task, he needs to select an Azure RMS template or use an Azure RMS AdHoc-Protection. In order to get the template information from Azure Information Protection, the user needs to be authenticated. Post successful authentication, the user can request templates and services, and return them. If all the previous steps are successful, the required template can be applied and the document will be encrypted. Furthermore, a functionality to unencrypt the document must be included in your design and development to store sensitive information which is not encrypted. Both encryption and unencryption are important to have full solution.

If we take a look at the Azure IP Test application you downloaded at the beginning of the chapter, in the *Technical requirements* section, you can start to build such an application with the help of this sample, because it provides many main functions of the SDK.

Otherwise, Microsoft provides other examples that help you start to gather more knowledge. A perfect way to start is through simple interaction with text files, which you can do with the `IpcNoteapp` application. This app shows you how you can protect a simple text file. My favorite way is to start directly with a specific use case, such as the protection of data in Azure Blob storage. If you think like me, start with `IpcAzureApp`, because this automatically introduces you to `IpcManagedAPI`, which provides you with many helper classes. Otherwise, if you want to explore more about the DLP functionality, you can use `IpcDlpApp`, which interacts with the File API to protect or unprotect data. You can monitor directories on your filesystem and apply protection policies with `RMSFileWatcher`. If you also want to see such functionality in a simple PowerShell script, you can use the following example:

1. Set the folder that you want monitor, including the subfolders:

```
$watcher = New-Object System.IO.FileSystemWatcher
 $w.Path = "C:\Users\jochen.nickel\Desktop\Monitored"
 $w.Filter = "*.*"
 $w.IncludeSubdirectories = $true
 $w.EnableRaisingEvents = $true
```

2. Define the actions for when an event is detected:

```
$action = {
 $fname = $Event.SourceEventArgs.Name
 if ($fname -like "*.tmp" -or $fname -like "*~$*"){}
 else {
 $path = $Event.SourceEventArgs.FullPath

 $changeType = $Event.SourceEventArgs.ChangeType
```

```
 $logline = "$(Get-Date), $changeType, $path"
 #write-host $logline
 Add-content "C:\Users\jochen.nickel\Desktop\log.txt" -
value $logline
 }
 }
$action1 = {
 $fname = $Event.SourceEventArgs.Name
 if ($fname -like "*.tmp" -or $fname -like "*~$*" -or
$fname -like "*.TMP" ){}
 else {
 $path = $Event.SourceEventArgs.FullPath

 $changeType = $Event.SourceEventArgs.ChangeType
 $logline = "$(Get-Date), $changeType, $path"
 write-host $logline
 Add-content "C:\Users\jochen.nickel\Desktop\log.txt" -
value $logline
 Set-AIPFileLabel $path -LabelId
'6523069d-6f4a-4486-8dc3-71de23457c71'
 $logline = "$(Get-Date), APPLIED: Set-AIPFileLabel $path
-LabelId '6523069d-6f4a-4486-8dc3-71de23457c71'"
 write-host $logline
 Add-content "C:\Users\jochen.nickel\Desktop\log.txt" -
value $logline
 }
 }
```

3. Decide which events should be monitored:

```
$created = Register-ObjectEvent $w "Created" -Action
$action
 $changed = Register-ObjectEvent $w "Changed" -Action
$action
 $deleted = Register-ObjectEvent $w "Deleted" -Action
$action
 $renamed = Register-ObjectEvent $w "Renamed" -Action
$action
 while ($true) {sleep 2}
```

To extend your knowledge, with the inclusion of the Azure **Active Directory Authentication Library** (**ADAL**), you can encrypt files with the `FormFileEncrypt` application. If you need to support HYOK scenarios, I recommend you work through `DualSererTestApp`. With your lab infrastructure, you're able to run every scenario.

For more information, check out the Microsoft developer reference (Azure Information Protection Developer's Guide): `https://bit.ly/2HIDjQ2`.

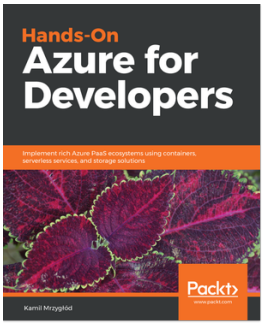
Summary

In this final chapter, we first introduced you to the several PowerShell scripts and the related code of the SDK to gather more information about the whole Microsoft Information Protection solution framework, or to start developing your extensions or applications. It was never the intention to turn you into a developer in one chapter, but using these resources is a good source of knowledge for troubleshooting or administering such a solution. You should now be able to describe which SDKs are available and what they can be used for. Furthermore, you have enough code samples to start working. We wanted to provide you with a functional and ready work environment to help you get started easily.

Hopefully, the information in this book will help you with your current organization or project. I want to thank you very much sticking around till the end of this book. If you have any questions, don't hesitate to ask me under my blog jochennickel.ch or info@inovit.ch.

Other Books You May Enjoy

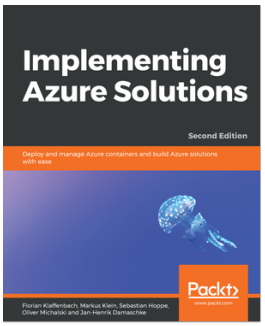
If you enjoyed this book, you may be interested in these other books by Packt:



Hands-On Azure for Developers
Kamil Mrzygłód

ISBN: 9781789340624

- Implement serverless components such as Azure functions and logic apps
- Integrate applications with available storages and containers
- Understand messaging components, including Azure Event Hubs and Azure Queue Storage
- Gain an understanding of Application Insights and other proper monitoring solutions
- Store your data with services such as Azure SQL and Azure Data Lake Storage
- Develop fast and scalable cloud applications



Implementing Azure Solutions - Second Edition
Florian Klaffenbach et al.

ISBN: 9781789343045

- Create and manage a Kubernetes cluster in Azure Kubernetes Service (AKS)
- Implement site-to-site VPN and ExpressRoute connections in your environment
- Explore the best practices in building and deploying app services
- Use Telemetry to monitor your Azure Solutions
- Design an Azure IoT solution and learn how to operate in different scenarios
- Implement a Hybrid Azure Design using Azure Stack

Leave a review - let other readers know what you think

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

O

P

S

T

U

V

W

Y