Microsoft

Extending identities to the cloud with Azure AD

The foundations of hybrid identity

External resources disclaimer

This material includes links to external publicly available articles, projects, and research papers which are provided to you as a convenience and for informational purposes only.

Microsoft bears no responsibility for the accuracy, legality, content or any other aspect of the external site. Use of external hyperlinks does not constitute an endorsement by Microsoft of the linked content.

The external content referenced in this document belongs exclusively to their respective author(s). Inclusion in this presentation does not grant you with any right on the external content. You must comply with the original source’s applicable policies.

How to use this document

Why this document?

This document is provided as a companion of the video lessons. Additional information is included here which would not fit the video format or would exaggeratedly lengthen the videos. As you are watching the videos, the instructor will point you to additional content in this document.

Structure

The structure of this slide deck follows the structure of the lessons. One slide deck is provided for each module. The slide deck has the same structure (naming of chapters and sections) than the video so that you can quickly jump to the slides associated with the lesson you're currently watching.

Foreword

This deck contains some design artefacts which all have their importance…

This sticky note icon is used to introduce the **abbreviation** of a concept or a technical word. Once the abbreviation has been introduced, the full version is no longer mentioned.

You will also find a list of all abbreviations at the end of the deck.

We were all young once. A section with this icon will basically tell you the **history** you might have missed by not working with the technology for the last 20 years.

It is not because you are new that you don't have to know how we got here!

Professor Useful will introduce some **tricky technical details** which might not seem relevant at first but could end up being really useful if you want to dig deeper in the technology.

This frame contains…

- Takeaways so important that we framed them

How to know the slide level

This deck contains 3 different content levels:

1. Regular level, the common slide
2. Advanced level, a slide with this indicator at the top left 

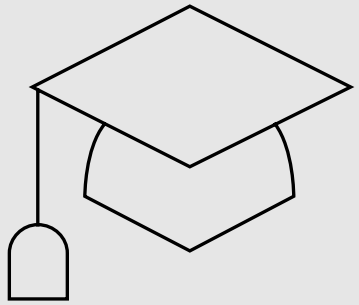
3. Additional content, all hidden slides

Section

2

Extending identities to the cloud with Azure AD

Learning Objectives



Describe the main components of an Azure AD environment integrated with AD

Agenda

1. Introduction of the main features of Azure AD
2. Integration with AD
3. Device integration in Azure AD
4. AD user authentication in Azure AD
5. Application integration in Azure AD
6. Administration of Azure AD

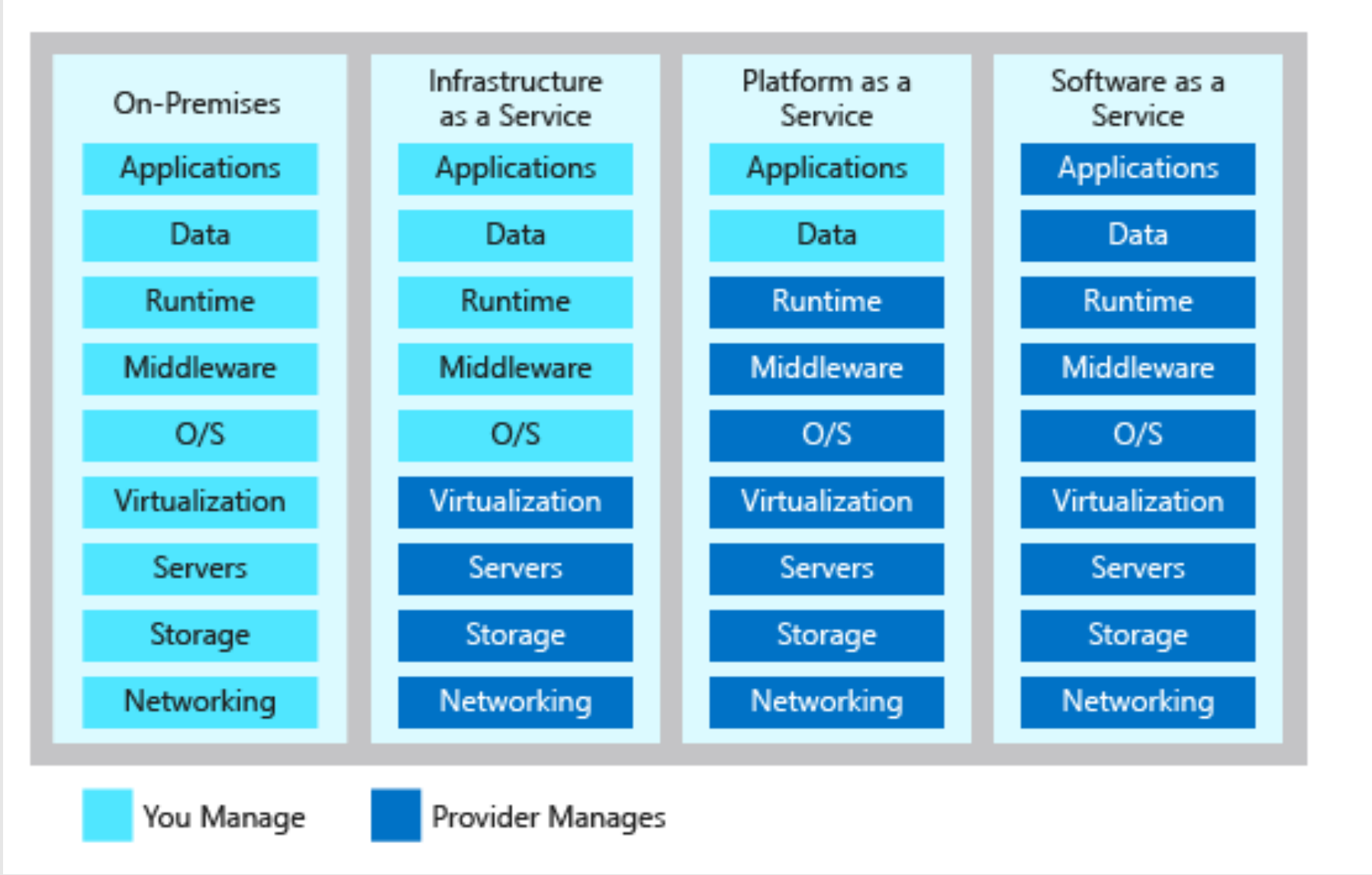
Chapter

1.2.1

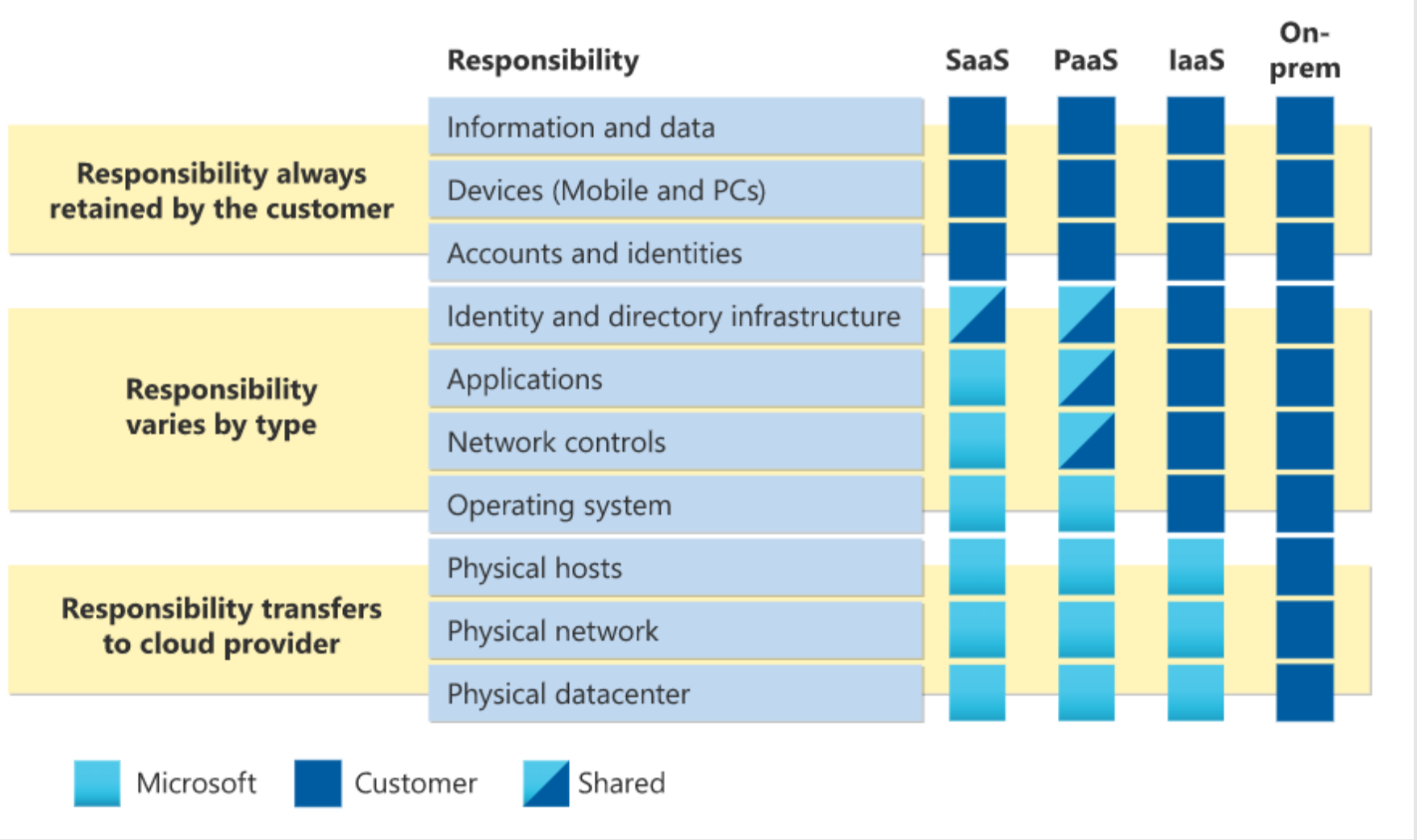
Introduction of the main features of Azure AD

🎯 Summarize the main features of Azure AD and differentiate between Azure AD and AD

IaaS – PaaS – SaaS

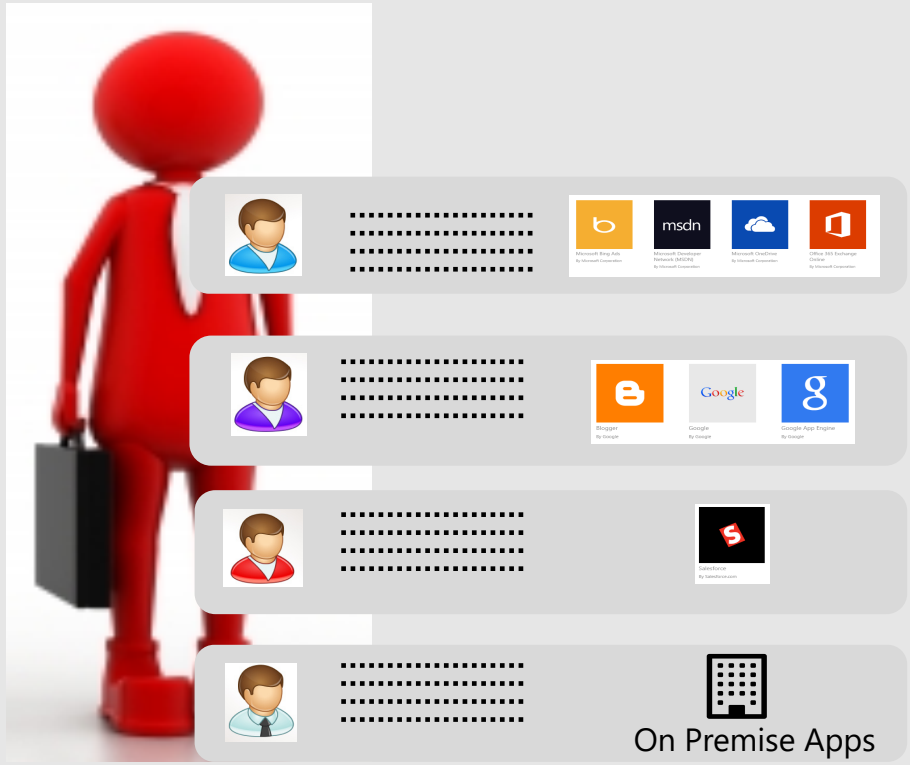


IaaS – PaaS – SaaS: Shared responsabilities



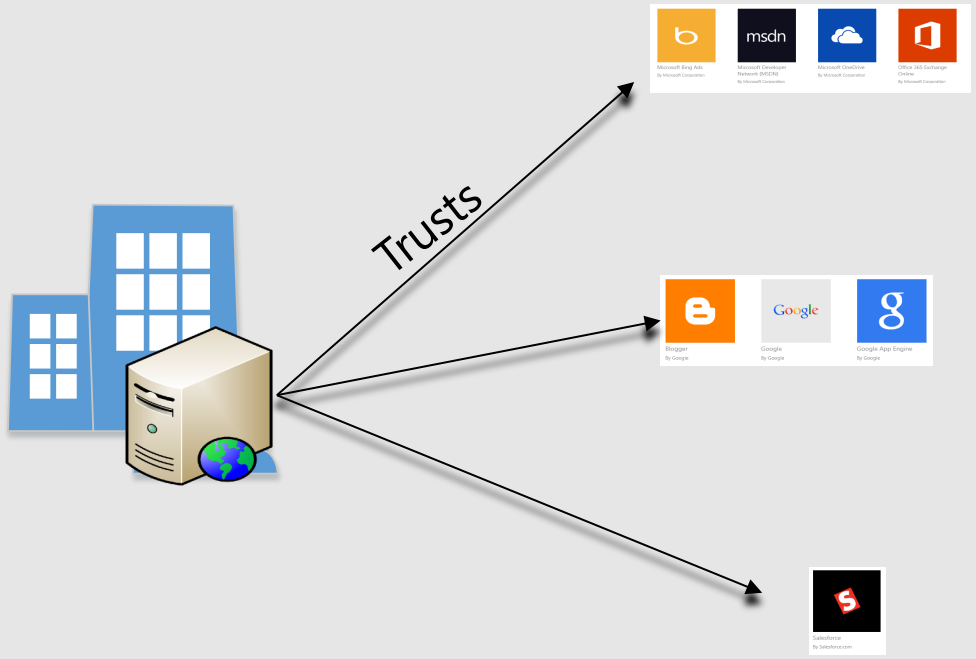
Adv.

IT Challenges: Multiple Identities/Trusts for Apps



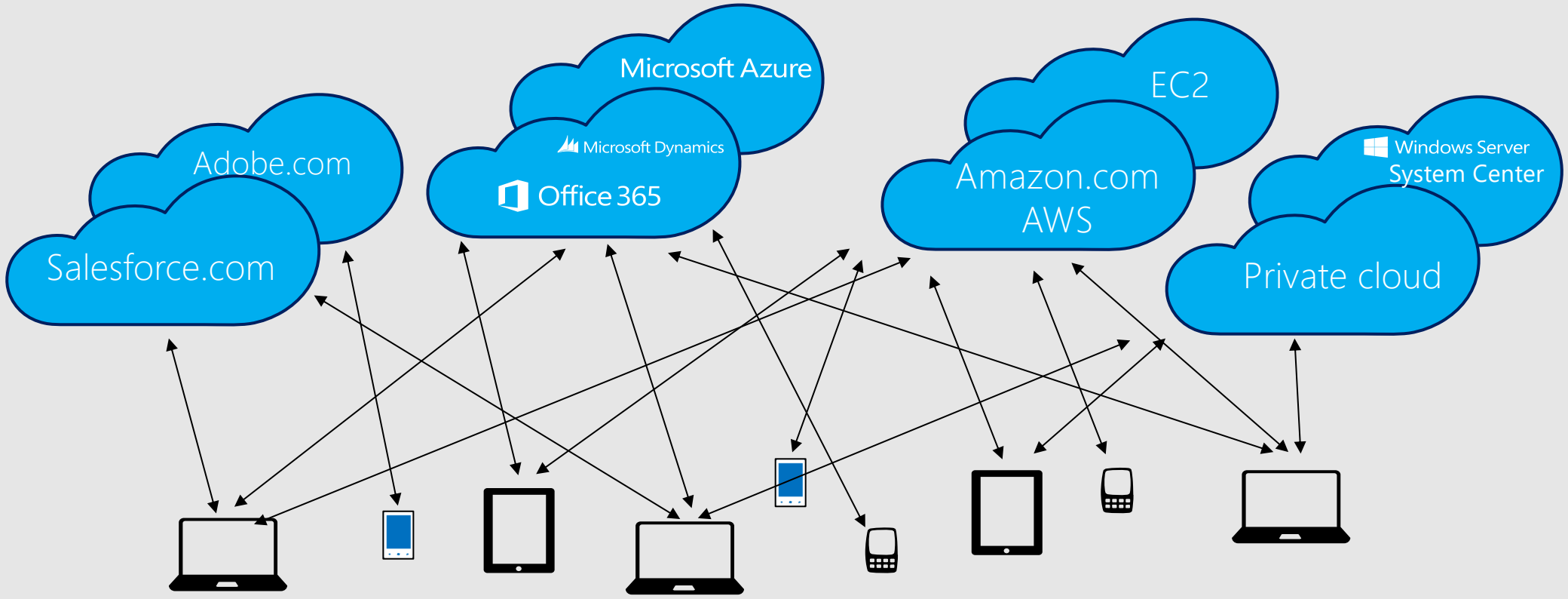
Create Separate accounts for each SaaS Apps

OR



Create separate federation trusts with SaaS Providers

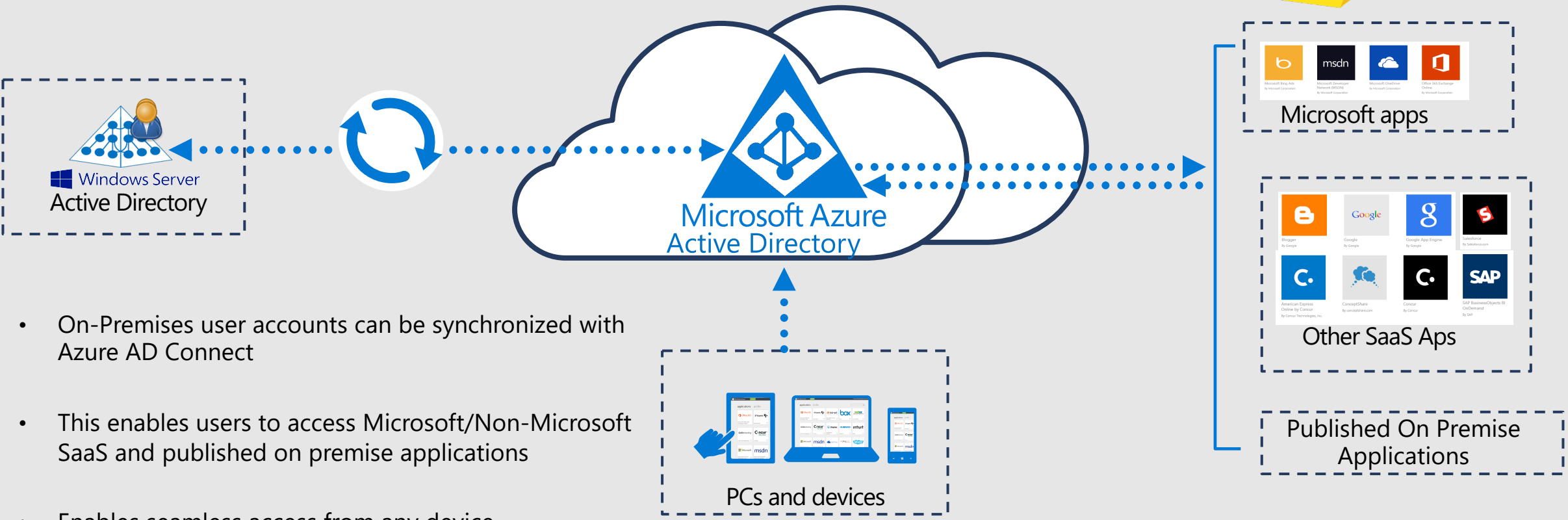
IT Challenges: Complicated Device access to Apps



- Manage Multiple Access Policies for each device
- Ensure security compliance of each device

Unified Identity and Access Management

IAM



- On-Premises user accounts can be synchronized with Azure AD Connect
- This enables users to access Microsoft/Non-Microsoft SaaS and published on premise applications
- Enables seamless access from any device
- Azure/On-Prem ADFS access control policies can be implemented for the devices
- Intune ensures the security compliance and effective MDM for the devices

What is a IDaaS?

An IDentity as a Service is cloud-based authentication service

No local infrastructure, all are provided, managed by the cloud provider

Improve security by using modern authentication and modern security features

- Like Multi Factor Authentication or Conditional Access

Enable SSO with other IDaaS or SaaS applications

Azure Active Directory

AAD

Azure Active Directory is a comprehensive identity and access management cloud-based solution that provides a robust set of capabilities to manage users and groups.

AAD is an Identity as a Service

IDaaS

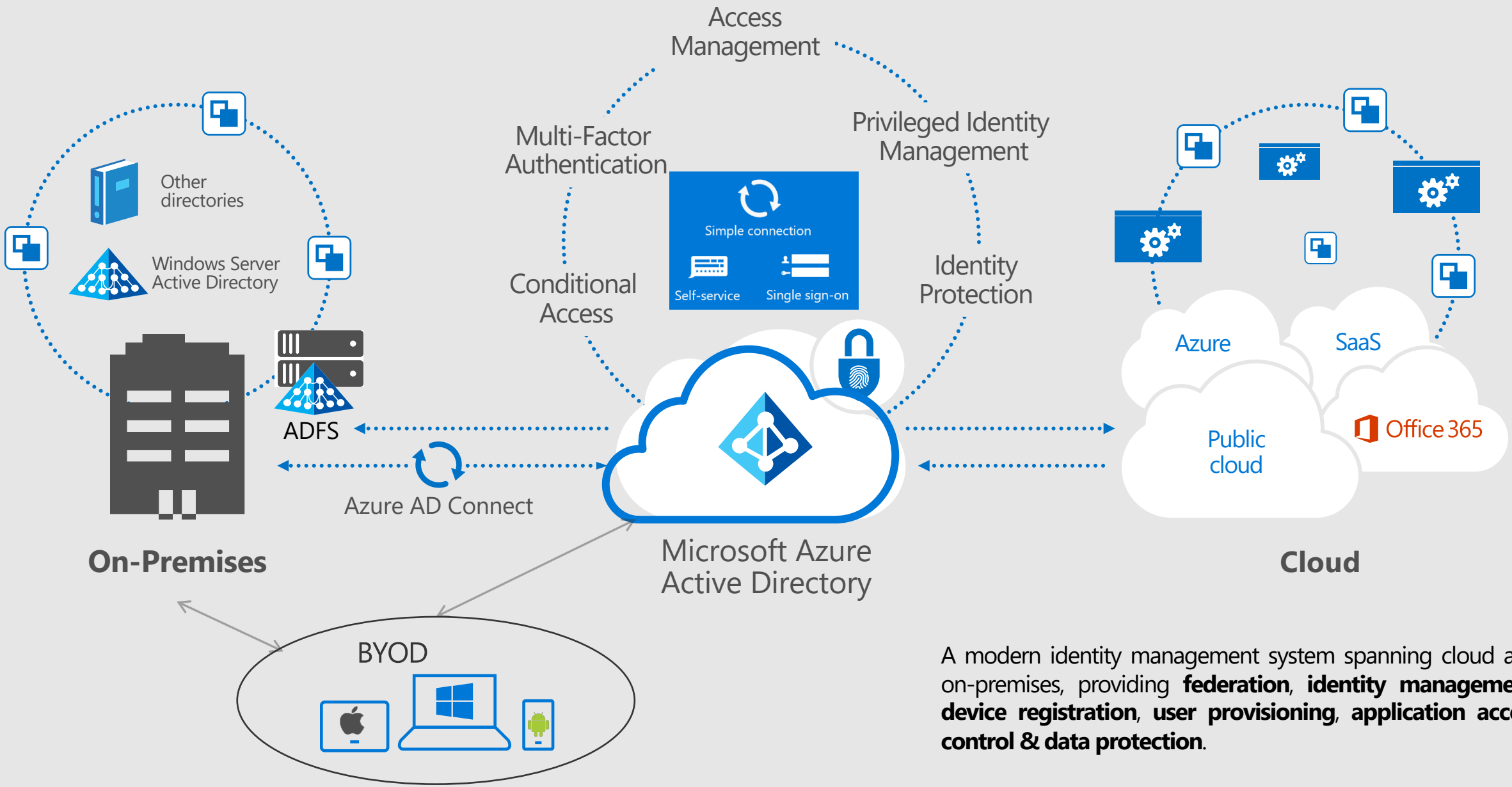
Azure AD is NOT the same as Windows Active Directory Domain Services.

AAD provides the core directory and identity management capabilities behind most of Microsoft's cloud services, including:

- Azure
- Microsoft Office 365

Provide a central solution to be authenticated with SSO with all App registered in the Tenant

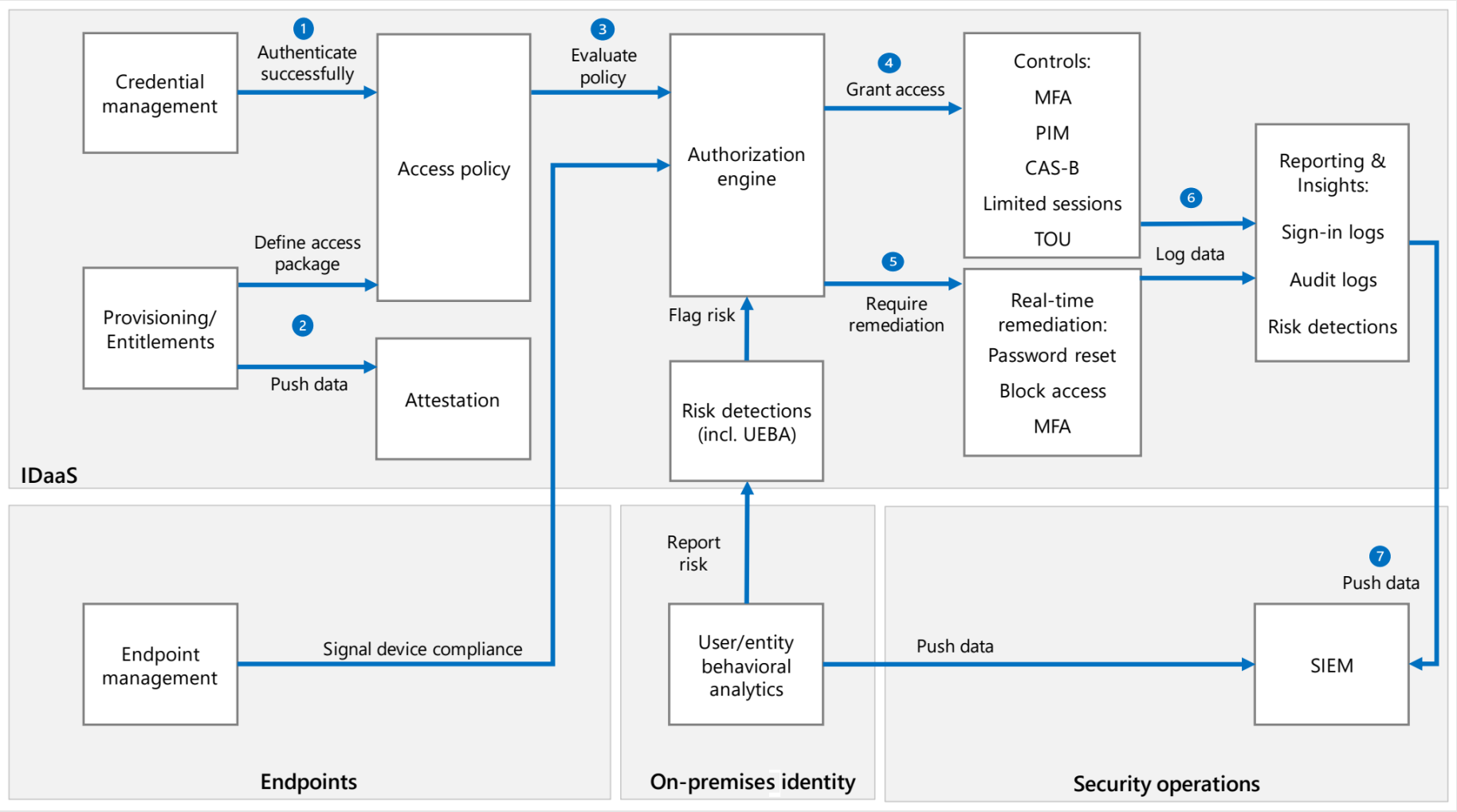
Azure AD: The Big Picture: Identity as the core of enterprise mobility



A modern identity management system spanning cloud and on-premises, providing **federation**, **identity management**, **device registration**, **user provisioning**, **application access control & data protection**.

Adv.

IDaaS a secure service by design



The difference between AD DS and Azure AD

Azure AD is a multi-customer public directory service, which means that **within Azure AD you can create a tenant** for your cloud servers and applications such as Office 365.

- Users and groups are created in a flat structure without OUs or GPOs.
- Authentication is performed through protocols such as SAML, WS-Federation, OAuth and OpenID Connect.
- To query Azure AD, you must use a REST API called AD Graph API not LDAP
  - These all work over HTTP and HTTPS.

Be aware: Azure Active Directory domain concept is different than the AD DS Domain

- In AD DS, domain administration (security via Forest) , storage, and trust boundary
- In Azure AD, domain is more synonymous with a DNS domain

AD DS vs Azure AD

What are they?

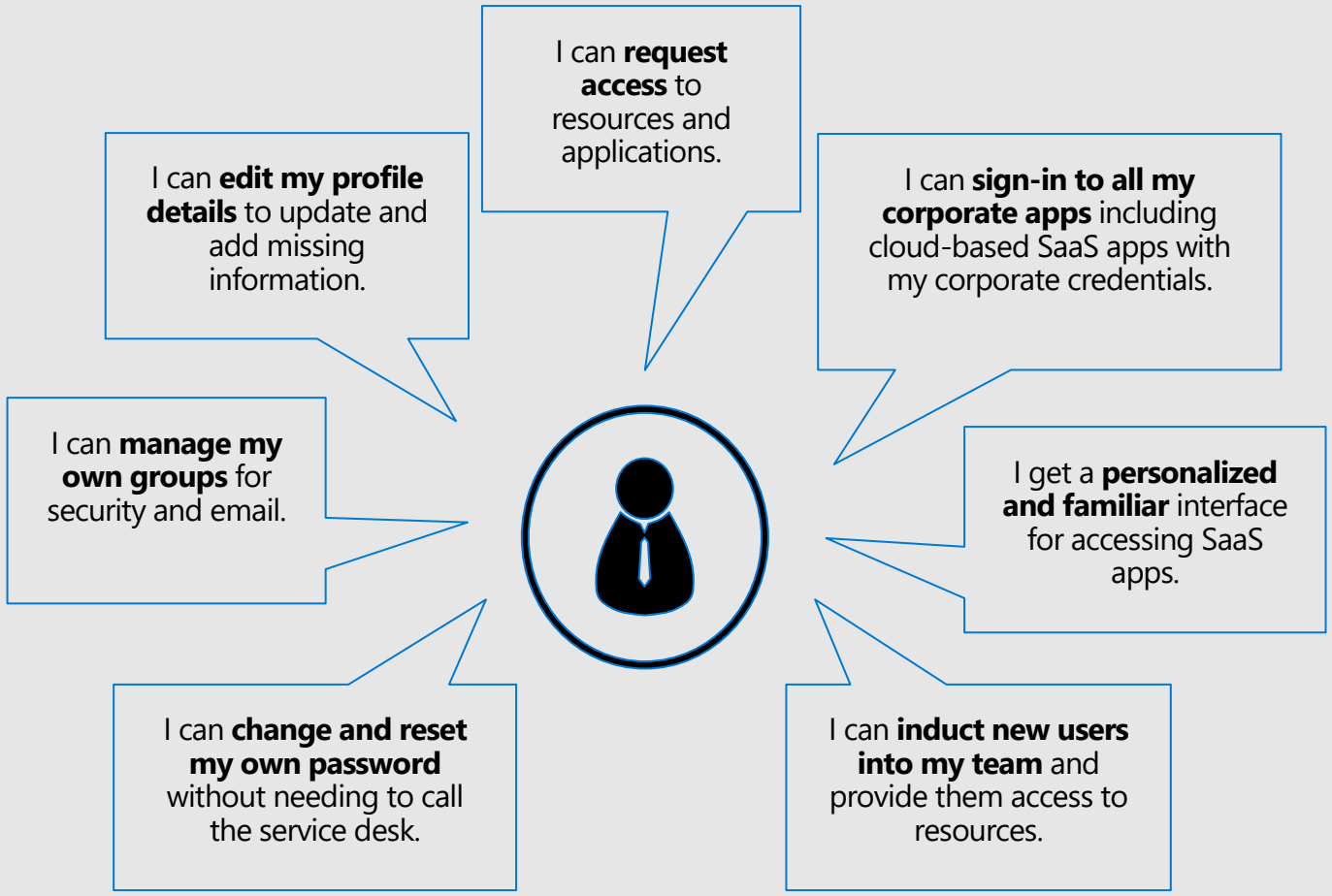
- Systems that store directory data and manage communication between users and resources
- Handle user logon processes, authentication, and directory searches

Service	AD DS	Azure AD
Runs on	Windows Server (physical, VM)	Microsoft multi-tenant cloud directory service
Structure	Based on X500/LDAP OU Structure	Flat (no OU structure)
Authentication Protocols	Kerberos, NTLM	SAML 2.0, OpenID Connect, OAuth 2.0, WS-Federation
Query Protocol	LDAP, DNS	AD Graph REST API
Focus	Authentication for on-prem resources, Group Policy	Authentication for cloud SaaS apps

Azure AD: Enabling End Users

Provides following features to enrich Identity and Access Management

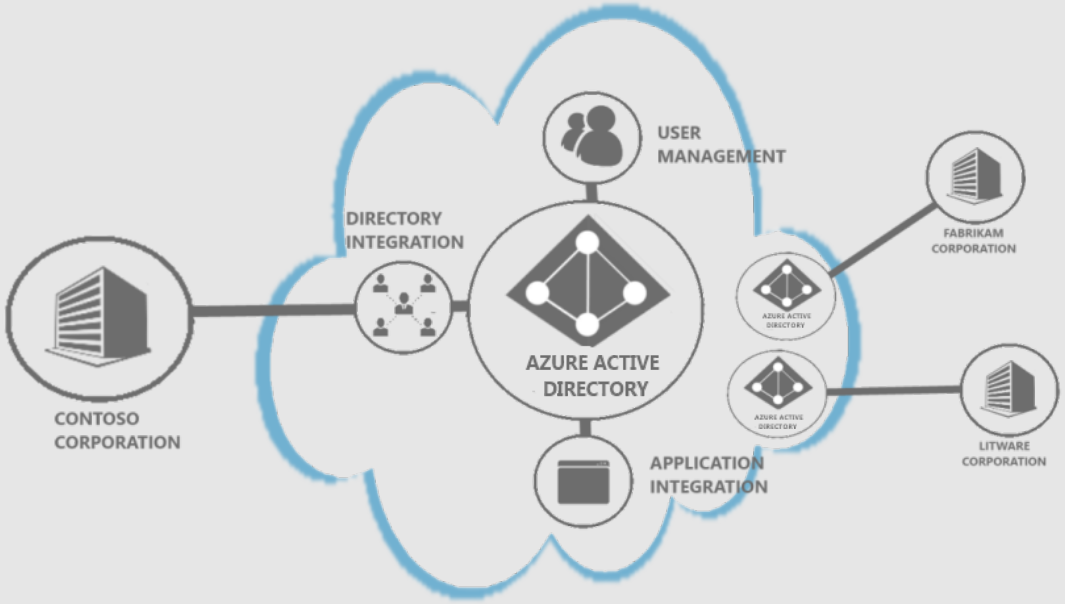
- Self Service Password Reset
- Multifactor Authentication
- Group Based Application Access
- Self Service Group Management
- Dynamic Group Management



What is an Azure AD Tenant?

In Azure Active Directory (Azure AD), a tenant is representative of an organization.

It is a dedicated instance of the Azure AD service that an organization receives and owns when it signs up for a Microsoft cloud service such as Azure, Microsoft Intune, or Office 365.



Tenant level isolation

Each Azure AD directory is distinct and separate from other Azure AD directories. This means that users and administrators of one Azure AD directory cannot accidentally or maliciously access data in another directory

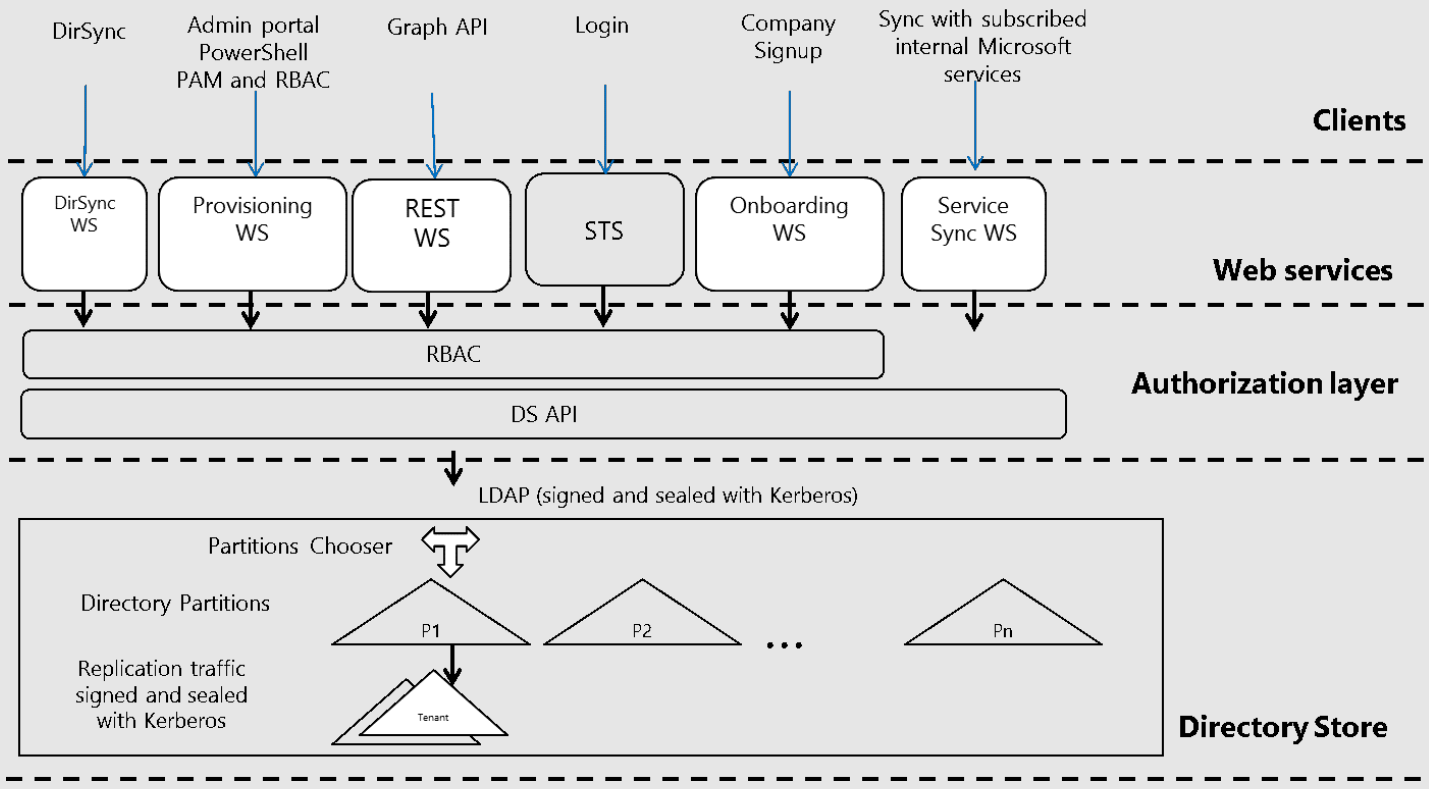
Tenant level isolation in Microsoft Azure is achieved using Azure Active Directory and role-based controls offered by it.

An Azure AD tenant is logically isolated using security boundaries so that no customer can access or compromise co-tenants, either maliciously or accidentally.

Tenant level isolation (Cont.)

Tenants are discrete containers and there is no relationship between them

No access across tenants unless tenant admin grants it through federation or provisioning user accounts from other tenants



Azure Active Directory: Namespaces

Namespace for any Azure/Office 365 tenant is

<TenantName>.OnMicrosoft.com

Custom domains can be added

- They are useful to avoid using the default tenant name (which is not known by users)
- Custom domains must be internet routable, and ownership needs to be verified

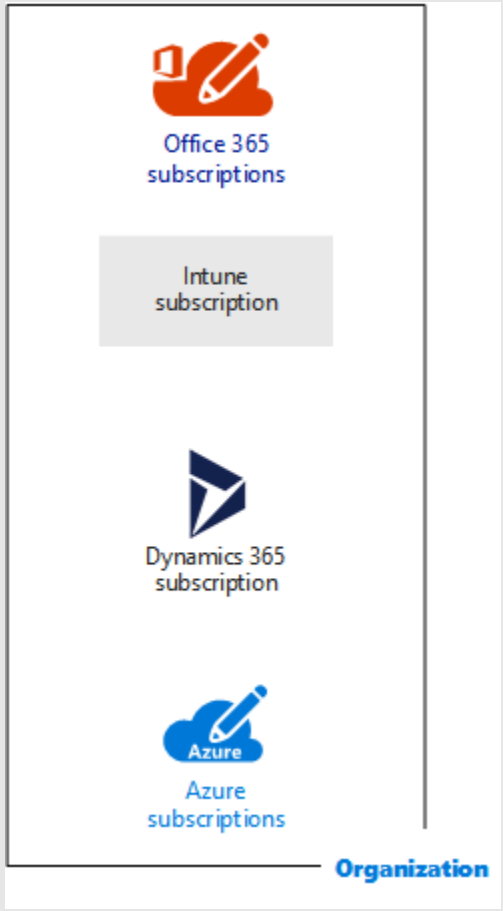
Some key concepts and differences

Organization (business entity) – Container for subscriptions

Subscriptions – like a Datacenter and more

An organization can have multiple subscriptions

- Example: for several department in the company
- A subscription can have multiple licenses
  - A subscription can be linked to a single Azure AD tenant
  - A tenant may be linked to multiple subscriptions
  - User accounts are stored in an Azure AD tenant



What! It's not free – AAD Editions

Azure Active Directory comes in three editions:

- Free
- Office 365 Apps
- Premium (P1, P2)

The Free edition is included with an Azure subscription, as well as other Microsoft cloud services including Office 365

The Premium editions (P1, P2) are available through a Microsoft Enterprise Agreement, the Open Volume License Program and the Cloud Solution Providers program

Azure and Office 365 subscribers can also buy Active Directory Premium online

Azure AD Editions: Common Features

Directory Objects

- Manage up to 500,000 objects in Free Edition. There's no limit for other editions.

User/Group Management/ User-based provisioning, Device registration

Single Sign-On (SSO)

Self-Service Password Change for **cloud** users

- Work for synchronized accounts but that requires higher editions of AAD

AD Connect (Sync engine)

Basic Security/Usage Reports

Azure AD Editions: Premium P1 Features

- Self-Service Group and app Management
- Self-Service application additions/Dynamic Groups
- Self-Service Password Reset/Change/Unlock with on-premises write-back
- Device write-back
- Azure AD Hybrid Password protection
- MIM CAL + MIM Server
- Cloud App Security
- Azure AD Connect Health
- Azure AD Application Proxy
- Conditional Access based on group, location and device state
- Azure Active Directory Join – Windows 10 only features

Azure AD Editions: Premium P2 Features

Identity Protection

Vulnerabilities and risky accounts detection

Risk events investigation

Risk based Conditional Access policies

Identity Governance

Privileged Identity Management (PIM)

Access Reviews

Entitlement Management

Azure AD deployment model

Azure AD allow to sync Users, Groups, Computers from AD (Onprem)

- It's **Hybrid Identity**

Azure AD allow 3 different deployment model:

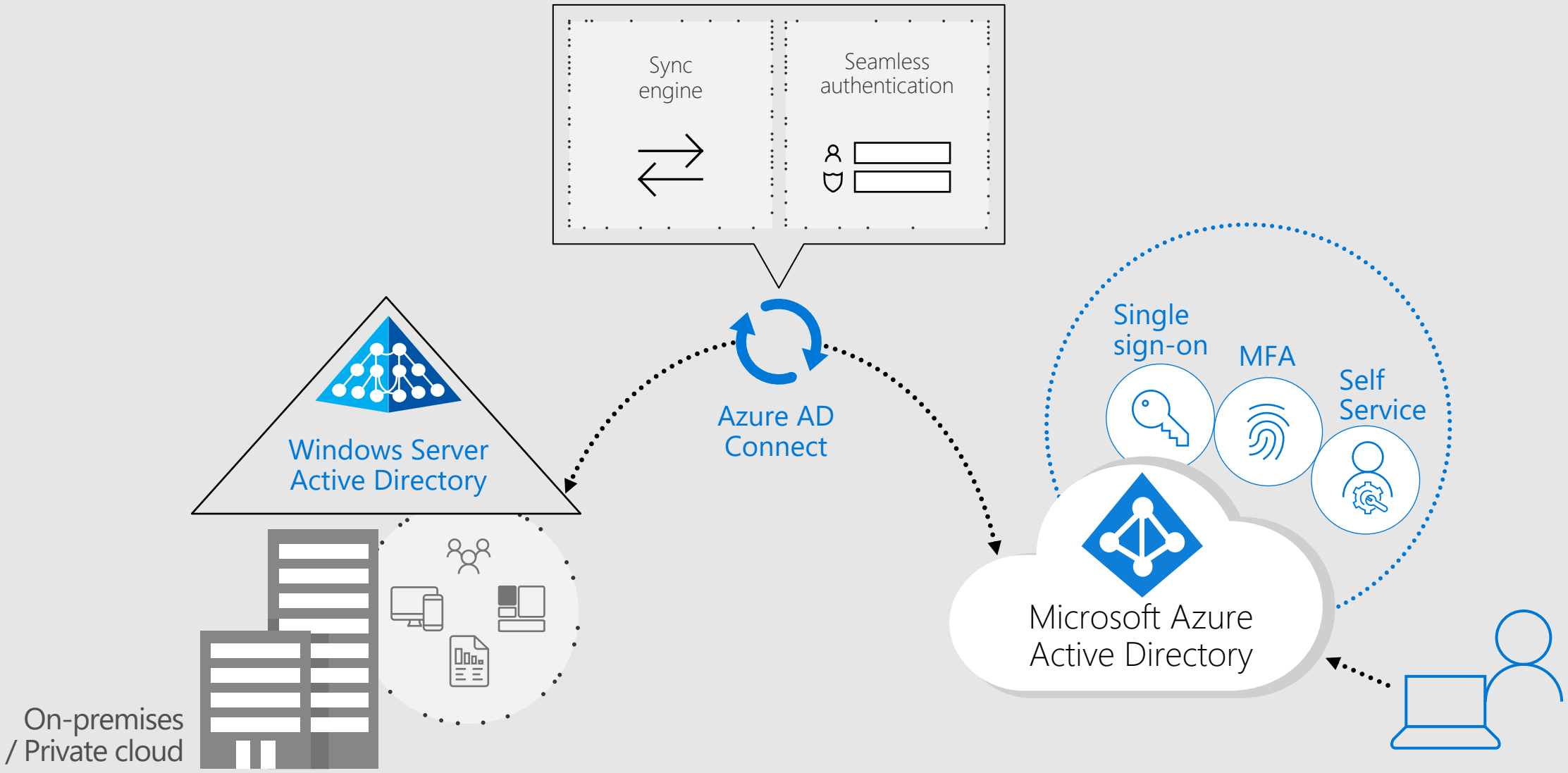
PHS

- Managed (Password Hash Sync)
  - The Hash (Hash (Password)) is synchronized from AD but it's AAD who authenticate the account
  - The simpliest

PTA

- Pass-through authentication
  - Agent relay authentication from AAD to AD. It's AD who authenticate the user
- Federation
  - AAD is federated to AD. It's AD who authenticate the user
  - The Hash (Hash (Password)) can be synchronized from AD but it's not mandatory. Recommended for DRP

Deployment overview – Hybrid Identity



Main AAD Objects

Member accounts:

- Cloud accounts
- Sync users from Active Directory

Guest accounts:

- B2B accounts from external directory (AAD, MSA, Google, ….)

Groups:

- Security
- Office 365 (specific group with Teams, One Drive,… resources attached)

Computers:

- Azure AD Registered
- Azure AD Joined
- Hybrid Azure AD Joined

Microsoft account vs Work or school accounts

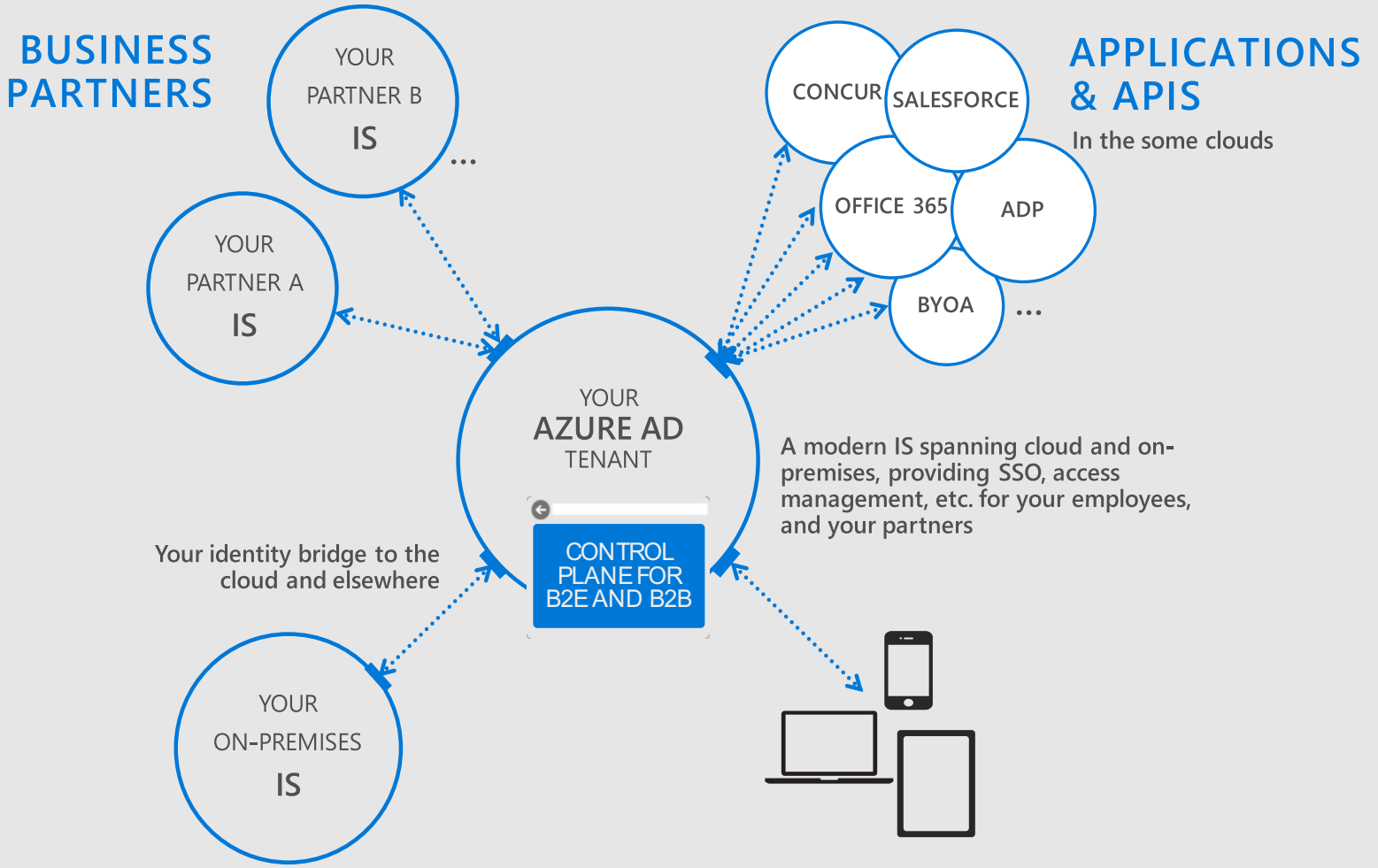
Microsoft account

- Created by you for personal use
- Provide access to consumer-oriented Microsoft products and cloud services
  - Like Outlook (Hotmail), OneDrive, Xbox LIVE, Personal Office 365 or any services link with Microsoft.
- These identities are created and stored in the Microsoft consumer identity account system run by Microsoft.

Work or school accounts

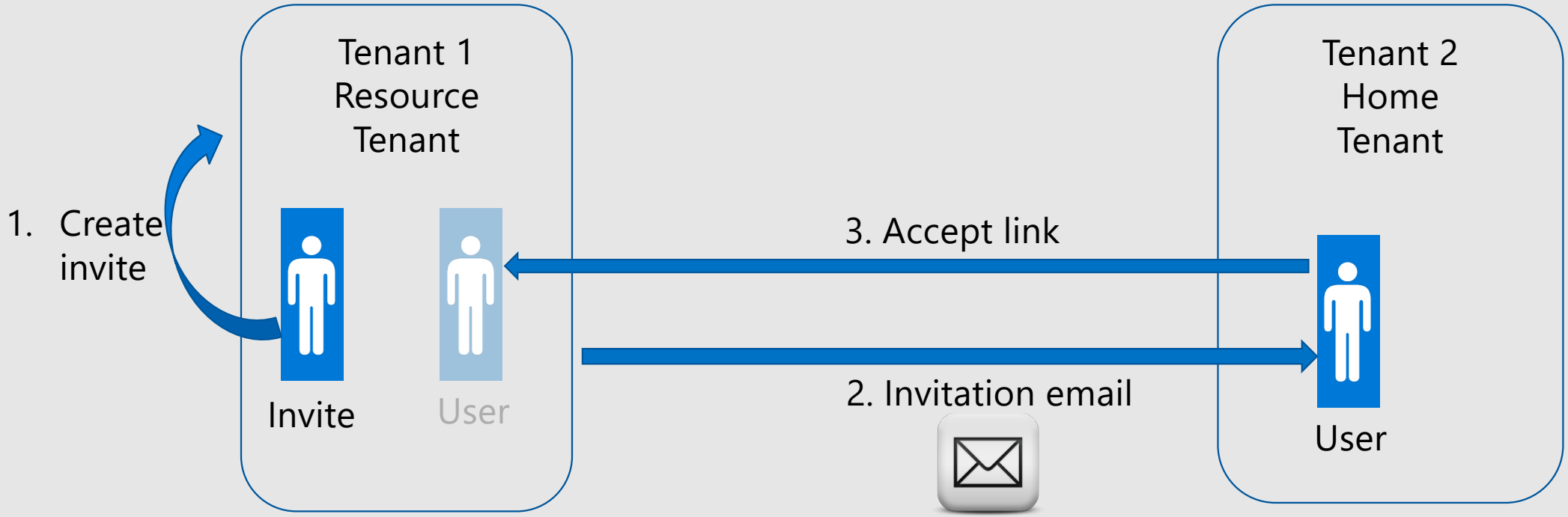
- Issued by an admin for business/academic use
- Provide access to enterprise business-level Microsoft cloud services
  - Like Azure, Intune, Office 365 or any Apps register in Azure AD

Extending Azure AD for external identities



Adv.

Business-to-Business (B2B) Users



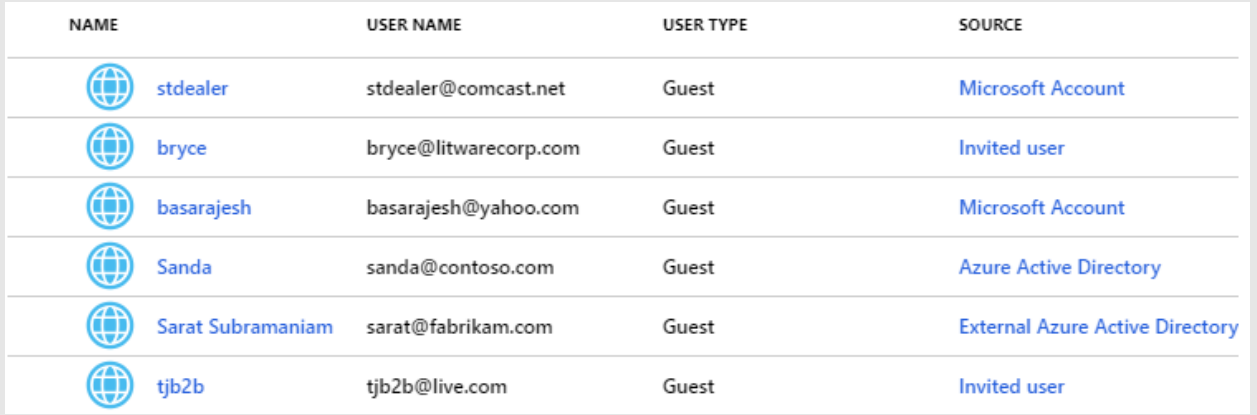
Key properties of a B2B user

UserType

This property indicates the relationship of the user to the host tenancy. This property can have two values:

- Member: This value indicates an employee of the host organization and a user in the organization's payroll. For example, this user expects to have access to internal-only sites. This user is not considered an external collaborator.

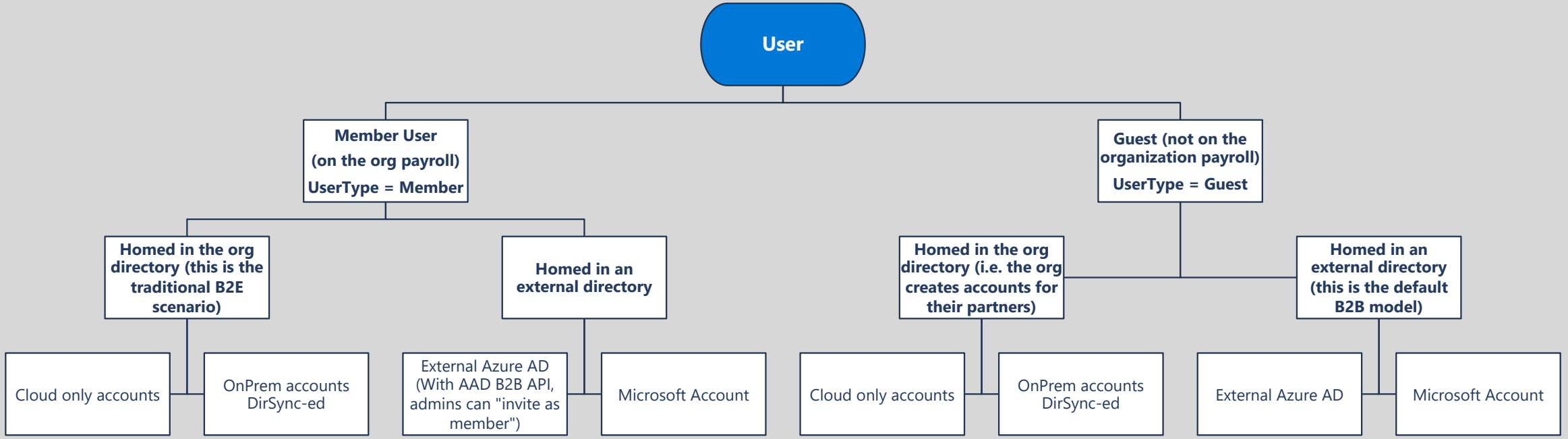
- Guest: This value indicates a user who isn't considered internal to the company, such as an external collaborator, partner, or customer. Such a user isn't expected to receive a CEO's internal memo or receive company benefits, for example.



NAME	USER NAME	USER TYPE	SOURCE
stdealer	stdealer@comcast.net	Guest	Microsoft Account
bryce	bryce@litwarecorp.com	Guest	Invited user
basarajesh	basarajesh@yahoo.com	Guest	Microsoft Account
Sanda	sanda@contoso.com	Guest	Azure Active Directory
Sarat Subramaniam	sarat@fabrikam.com	Guest	External Azure Active Directory
tjb2b	tjb2b@live.com	Guest	Invited user

Key properties of a B2B user

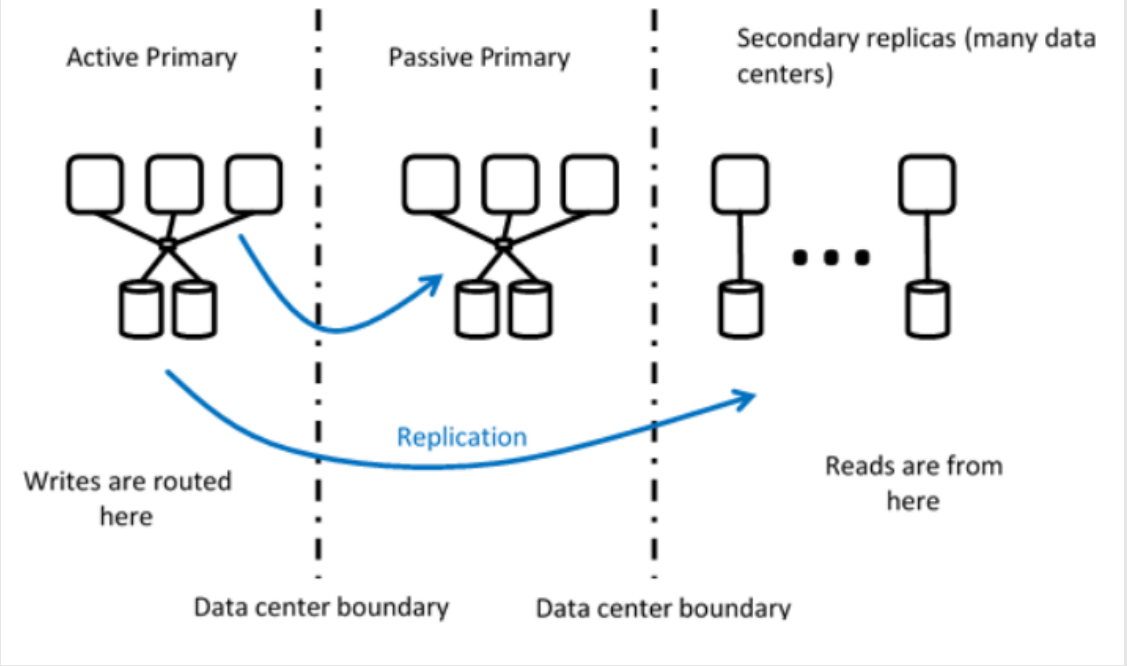
Depending on the inviting organization's needs, an Azure AD B2B collaboration user can be in one of the following account states:



Azure Active Directory Design

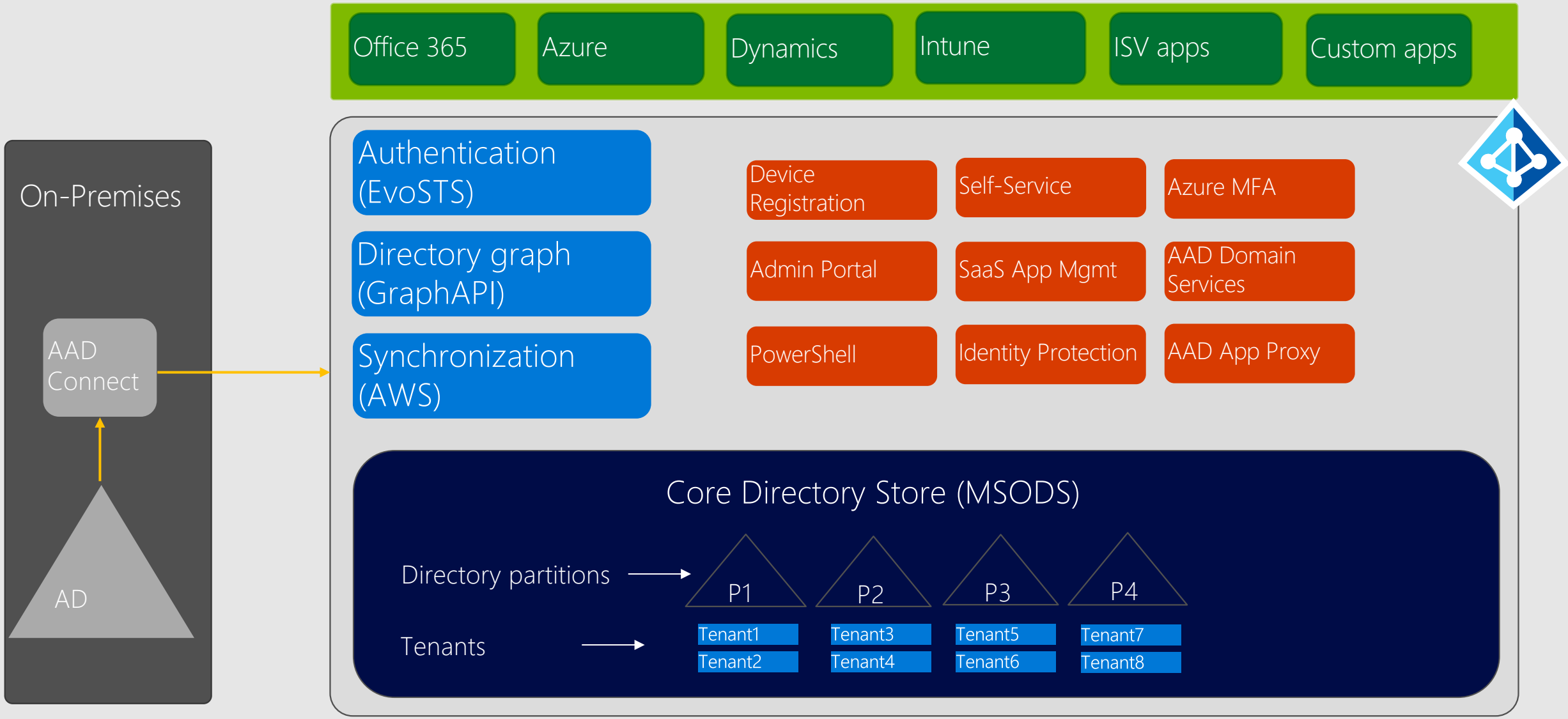
AAD works as a system of an active primary, a passive primary, and then several secondary replicas that are replicated around the world.

- Partitioned for scalability
- Active/active across data centers for directory reads and authentication (token issuance)
- Active/passive across data centers for directory writes
- Recovery Time Objective (RTO)
  - Zero for reads and token issuance
  - About five minutes for writes



Adv.

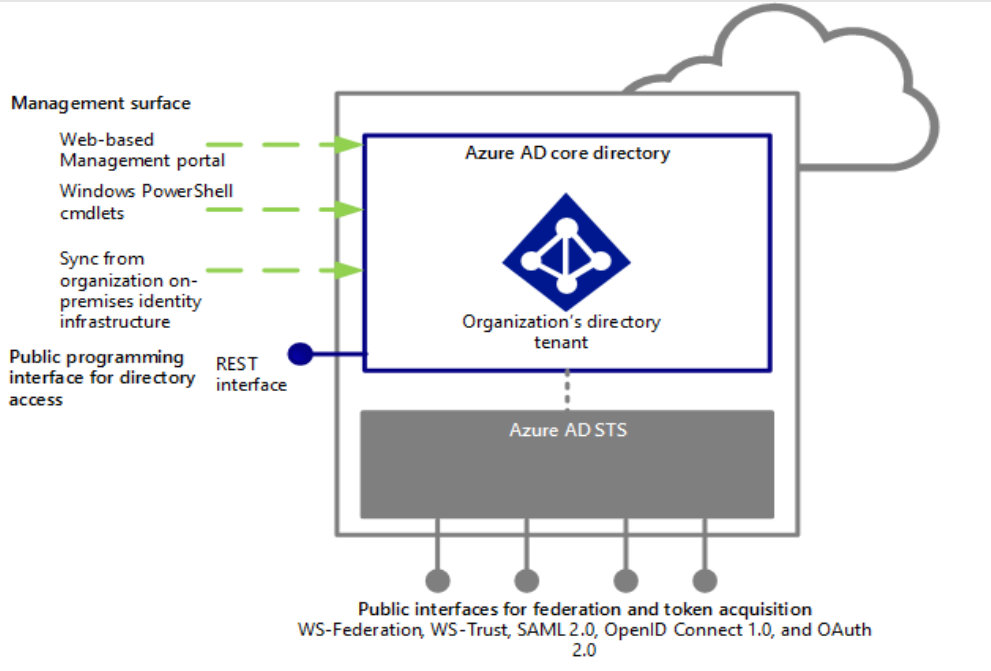
Azure AD internal architecture overview

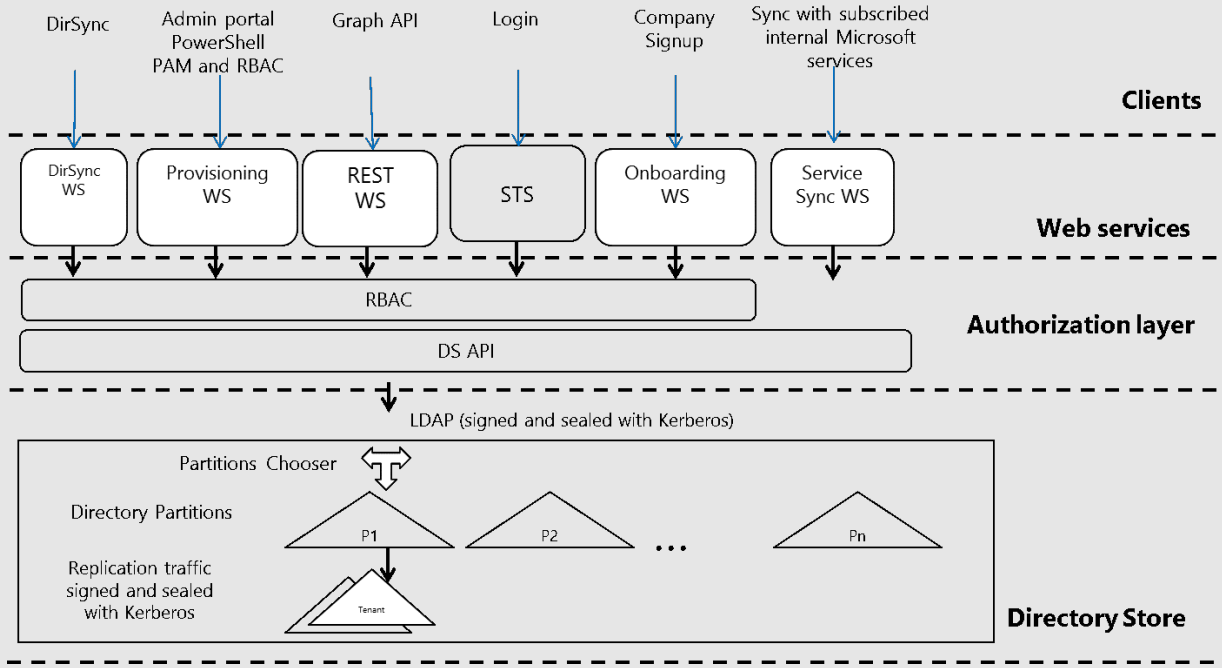


Anatomy overview of Azure AD

At the simplest level, Azure AD is a scalable directory infrastructure that provides for:

- storage,
- data access,
- replication and synchronization,
- device registration,
- Authorization & Authentication (via its security token services (STS) sub component)





Standards based integration

Azure Active Directory

- OAuth2 & OpenID Connect
- FIDO 2.0
- REST based Graph API (OData 4.0 Compliant)
- SAML
- WS-Federation
- SCIM 2.0

WS-Federation

Strong support for modern, cross platform, cloud friendly API's & protocols

Certification program for third party federation servers & services

Actively engaged in standards bodies (OAuth, OpenID, SCIM, etc.)

Chapter

1.2.2

Integration with AD

🎯Identify the components involved in identity hybridization

What's Hybrid Identity?

It's not a X-Men 😁

It's a common user identity for authentication and authorization to all resources, regardless of location

- Azure AD Connect is used to sync AD identity to Azure AD

To achieve hybrid identity with Azure AD, one of three authentication methods can be used, depending on your scenarios:

- **Password hash synchronization (PHS)**
- **Pass-through authentication (PTA)**
- **Federation (AD FS)**

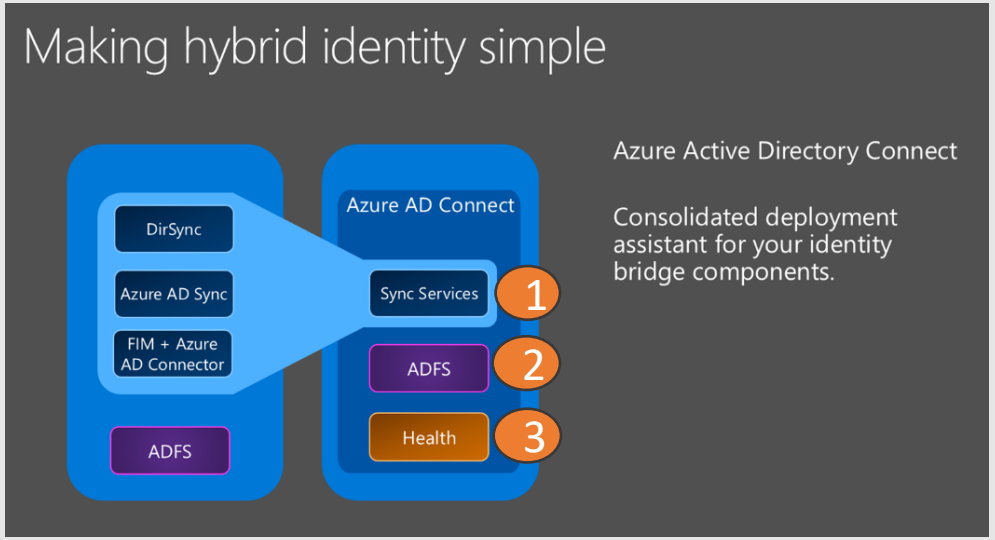
Azure AD Connect concept

- Azure AD Connect is an on-premises Microsoft application that's designed to meet and accomplish your hybrid identity goals
- Azure AD Connect provides the following features:
  - Password hash synchronization
  - Pass-through authentication
  - Federation integration
  - Synchronization
  - Health Monitoring

Azure AD Connect

AADC

3 main sub-components:

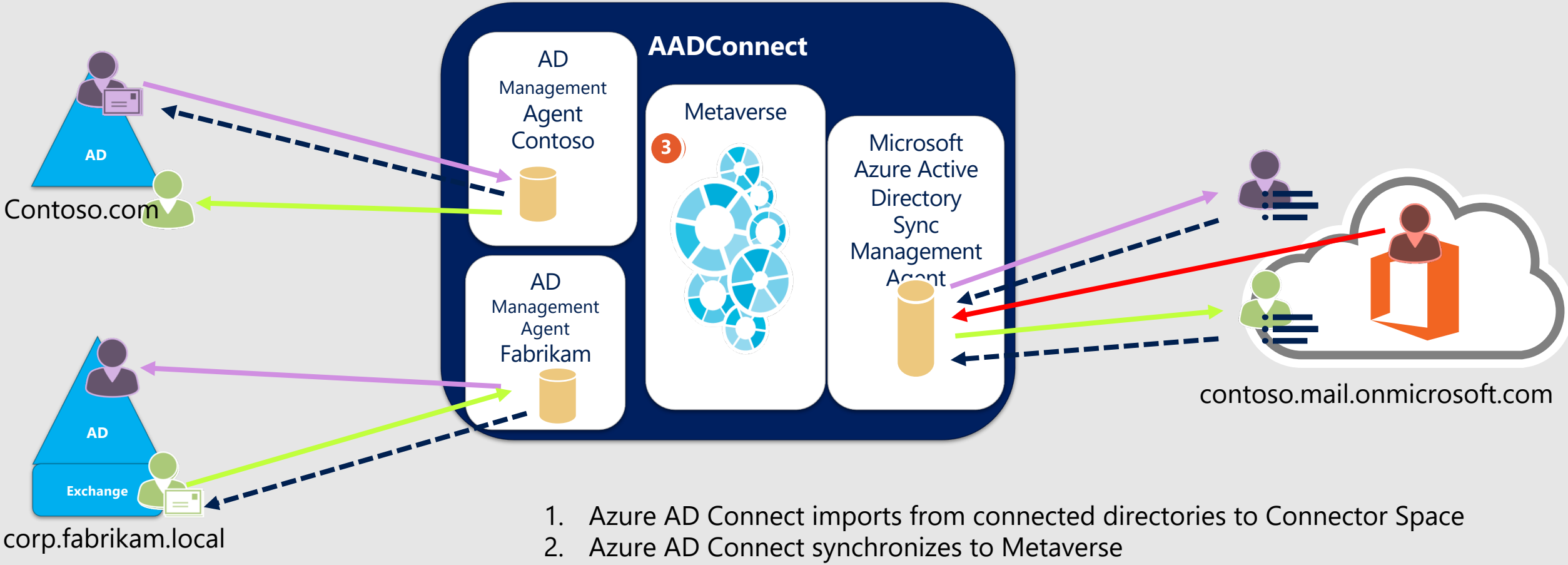


1. Identity Sync engine
   - Create users, groups and devices sync attributes to the cloud, write-back attributes/objects
   - Match identity information from on-premises and the cloud
   - Synchronize Password Hashes, write-back passwords
2. AD FS - Federation is an optional part of AADC
   - Azure AD Connect can Deploy and/or configure AD FS
3. Health Monitoring
   - Provide health monitoring for both Azure AD Connect (“The Sync”) and AD FS

The attributes and how they map are referred to as metadata

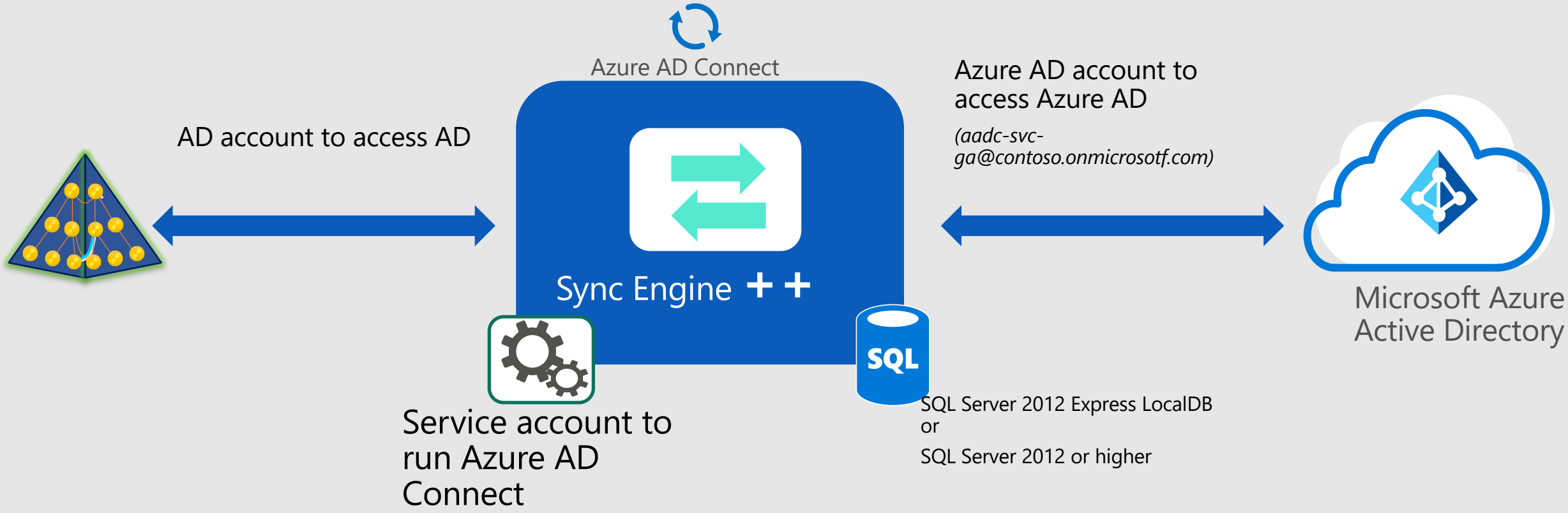
The metaverse shows how all of the identities are connected

Synchronization Process



1. Azure AD Connect imports from connected directories to Connector Space
2. Azure AD Connect synchronizes to Metaverse
3. Azure AD Connect uses attribute mapping to determine the type of object to provision or update
4. Azure AD Connect exports and/or revises the objects to the connected directories

Azure AD Connect components



Continuously evolving product: Automatic upgrades are possible (Set-ADSyncAutoUpgrade)

Azure AD Connect is a tool that's been evolving at a fast pace.

It's challenge to keep this content up to date/complete. Please refer to the online documentation for the latest and most accurate information.

Don’t Forget to secure your AADC!

Protect the AADC server as a Domain Controllers

- It’s a T0 Server, only allow Privileged Account to logon
- Use Security Compliance Tool to apply default Security Baseline at least

Protect the service account as a Privileged Account

- Can be used to create Golden Tickets (has the ACE Replicate Directory Change All)

Protect the server hosting the database

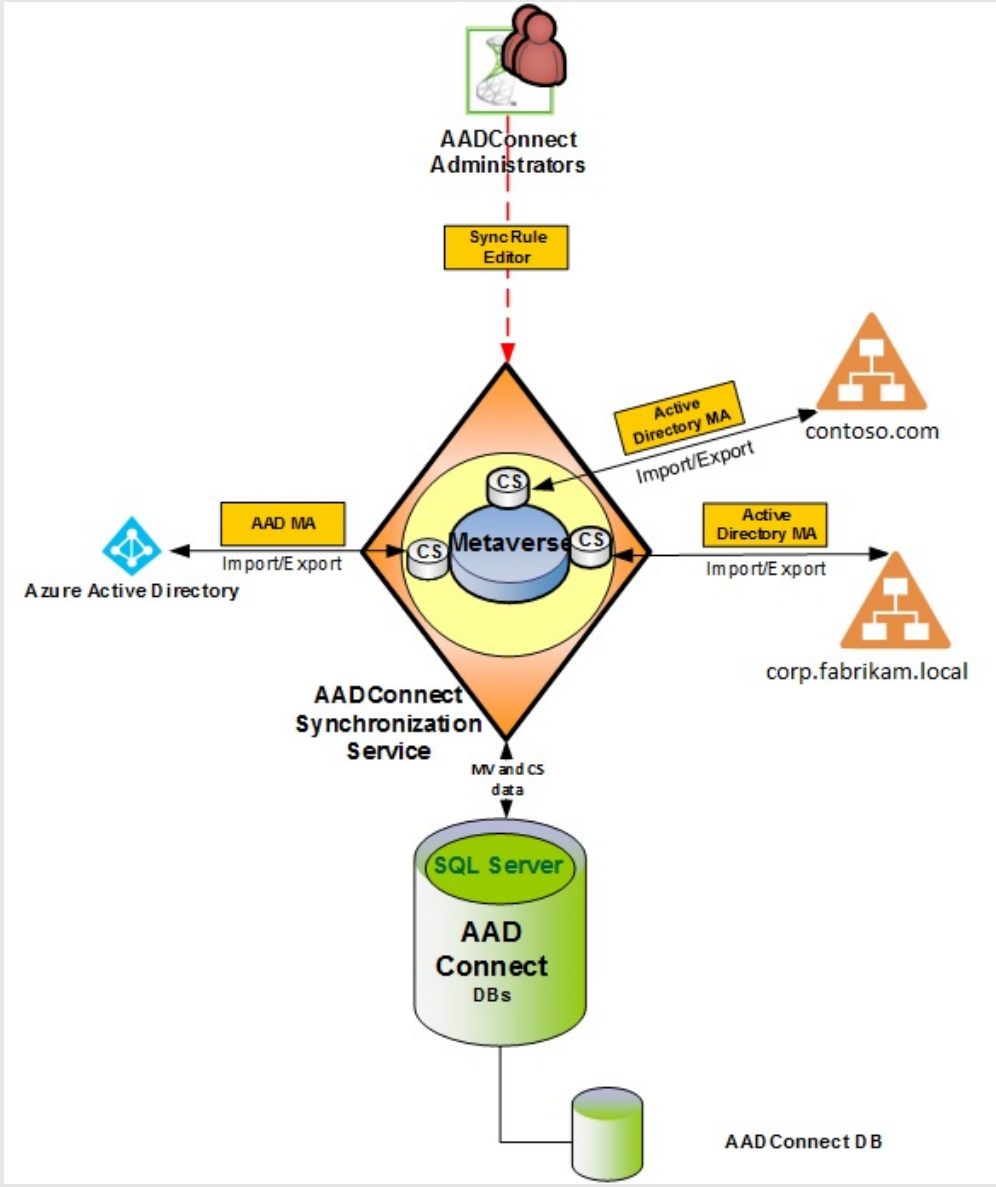
Azure AD Connect Concepts

Metaverse

- A metaverse is a consolidated view of all the joined identities from neighboring connector spaces
- All the data that is stored in an object must be contributed through an attribute flow. The metaverse maintains persistent connectors with every connector space

Connector Space

- Connector between the Directory and AADC
- Synchronize data between the directory and the metaverse



Choosing a Topology

AADC can be implemented in different topology

- Single forest
- Multiple Forests, Single Azure AD Directory
- Multiple Forests – Separate Azure AD Directories
- Multiple Forests – Match Users – Full Mesh with Optional GALSync
- Multiple Forests – Match Users – Account-Resource Forest

Staging Mode

The staging mode can be used in several scenarios, including:

- High availability
- Test and deploy the new configuration changes
- Introduce a new server and decommission the old one

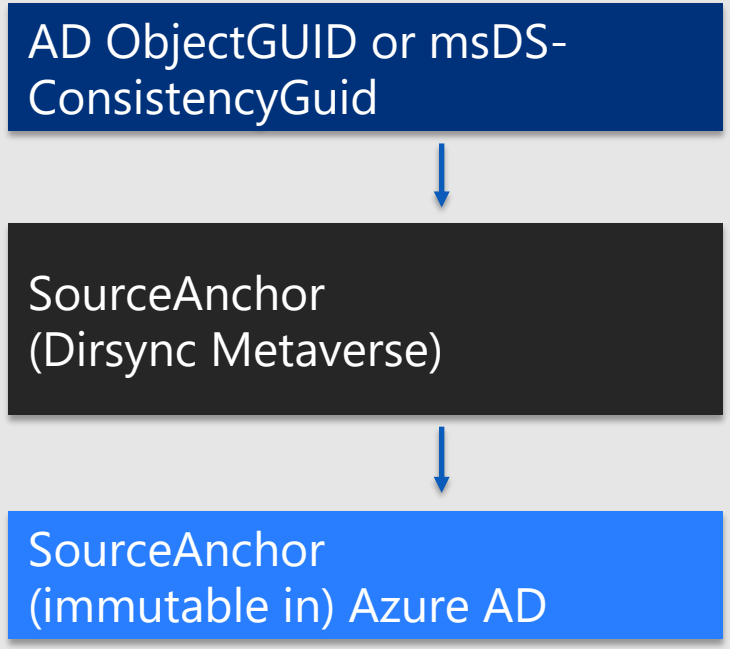
Azure AD Connect / key object “properties”: Source Anchor

It’s important to understand SourceAnchor (aka ImmutableID):

- Immutable in Azure AD
  - Used as the immutable identifier for any given object that is synchronized between on-premises and Azure AD
- Unique identifier based on AD ObjectGUID (by default!)
- Basis for the relationship between AD DS objects and Azure AD objects.

AAD Connect defaulted to using ObjectGUID

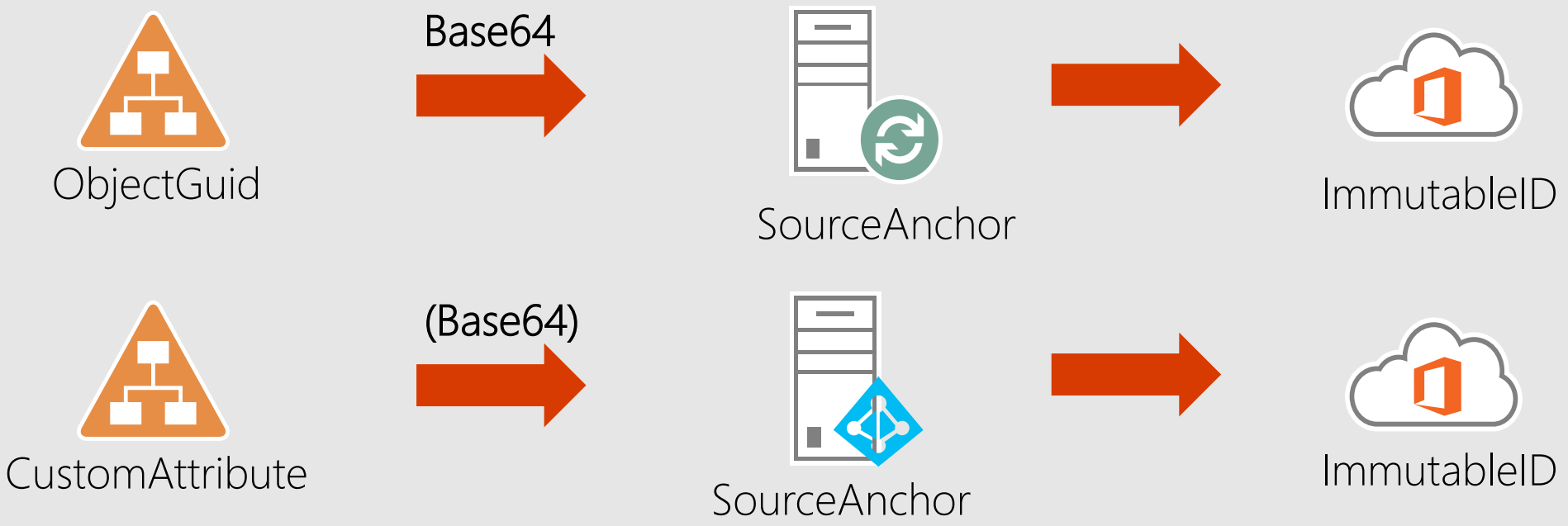
- AAD Connect uses msDS-ConsistencyGuid
- AAD Connect writes its objectGUID value to the msDS-ConsistencyGuid attribute
  - * Only if msDS-ConsistencyGuid is NULL
- Allows a user to move between forests without creating a duplicate in Azure AD
- We refer to this as the “ConsistencyGUID”



ImmutableID/SourceAnchor

im·mu·ta·ble [ih-myoo-tuh-buhl]
not mutable; unchangeable; changeless. Not subject or susceptible to change

```
FirstName                                   : Morten
ImmutableId                                 : WJFNmN+kW0CCOEpKnLCdZA==
```



Azure AD Connect: Source of Authority

Refers to the location where Active Directory objects are mastered (AD on-premises or Azure AD)

Activating and installing AAD Connect makes the on-premises Active Directory the source of authority

Once enabled, changes to objects replicated to Azure AD can only be made on-premises

Deactivating AAD Connect (in the Admin portal) transfers the source of authority back to the Azure Active Directory

Azure AD Connect and Write-back (options)

Password write-back

- Provide users Self Service Password Reset (SSPR) capabilities
- Ability to reset passwords on-prem. (Password sync and Federation customers supported)

Group write-back

- Office 365 groups written back as read-only DLs
- Requires Exchange schema
- Improves productivity in Hybrid scenarios (O365 groups appear in on-premises GAL)

Device write-back

- Supports ability to do conditional access in AD FS based on registered devices
- Share Azure DRS registered devices with (on-premise) Active Directory so that AD FS can leverage device authentication for these devices

Exchange Hybrid write-back

AADConnect Run Profiles

Full Import:

- Performs a full import, but does not synchronize the changes with the metaverse
- Objects are staged to the connector space

Delta Import

- Imports only the changes
- Does not synchronize changes with the metaverse

Full Synchronization

- Synchronizes all the objects with the metaverse, including the objects that have not changed

Delta Synchronization

- Synchronizes only the changes (deltas) with the metaverse

Export

- Exports the changes to the target system

AADConnect Filtering

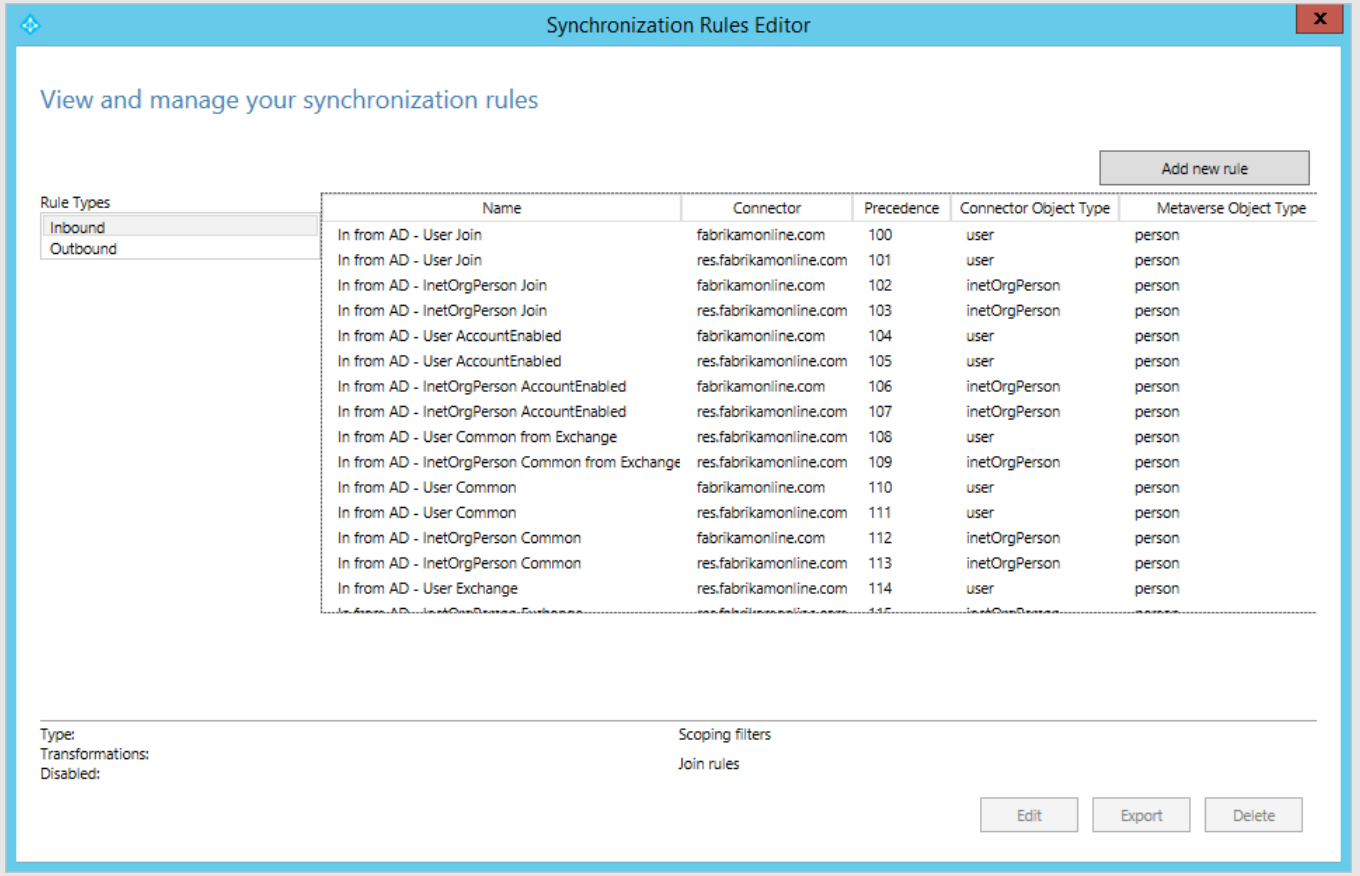
- **Group-based**
  - Filtering based on a single group can only be configured on initial installation by using the installation wizard.
- **Domain-based**
  - By using this option, you can select which domains synchronize to Azure AD. You can also add and remove domains from the sync engine configuration when you make changes to your on-premises infrastructure after you install Azure AD Connect sync.
- **Organizational unit (OU)–based**
  - By using this option, you can select which OUs synchronize to Azure AD. This option is for all object types in selected OUs.
- **Attribute-based**
  - By using this option, you can filter objects based on attribute values on the objects. You can also have different filters for different object types.
- **Using Synchronization Rules**

Synchronization Rules

To sync object, AADC use Synchronization Rules

AADC have default Inbound and Outbound rules to sync main objects:

- Users
- Groups
- Contacts
- Computers



Synchronization Rules

Synchronization Rule Name	Comment
In from AD – User Join	Rule for joining connector space objects with metaverse
In from AD – UserAccount Enabled	The attributes that are required for signing in to Azure AD and Office 365. We want these attributes from the enabled account
In from AD – User Common from Exchange	The attributes found in the Global Address List. We assume the data quality is best in the forest where we have found the user's mailbox
In from AD – User Common	Attributes found in the Global Address List. In case we did not find a mailbox, any other joined object can contribute the attribute value
In from AD – User Exchange	Will only exist if Exchange Server has been detected. Will flow all infrastructure Exchange attributes
In from AD – User Lync	Will only exist if Lync has been detected. Will flow all infrastructure Lync attributes

Adv.

Synchronization Rules

Users Filtered by Default (Out to AAD)

- Must have a SourceAnchor
- After the object has been created in Azure AD then sourceAnchor cannot change. If the value is changed on-premises, the object will stop synchronizing until the sourceAnchor is changed back to its previous value
- Must have the accountEnabled (userAccountControl) attribute populated. With an on-premises Active Directory, this attribute will always be present and populated

Adv.

Synchronization Rules

Users **Not Synchronized** by Default (In from AD)

IIF(IsPresent([isCriticalSystemObject])

IsPresent([sAMAccountName]) = False

[sAMAccountName] = "SUPPORT_388945a0“

Left([mailNickname], 14) = "SystemMailbox{"

Left([sAMAccountName], 4) = "AAD_"

(Left([mailNickname], 4) = "CAS_" && (InStr([mailNickname], "}") > 0))

(Left([sAMAccountName], 4) = "CAS_" && (InStr([sAMAccountName], "}") > 0))

Left([sAMAccountName], 5) = "MSOL_"

CBool(IIF(IsPresent([msExchRecipientTypeDetails]),BitAnd([msExchRecipientTypeDetails],&H21C0 7000) > 0,NULL))

CBool(InStr(DNComponent(CRef([dn]),1),"\\0ACNF:")>0), True, NULL)

[adminDescription] NOTSTARTWITH User_

Password write-back – allow the Self-Service Password

A cool feature for the User! It's Self-Service Password

- Allows users in your organization to reset their passwords automatically without calling an administrator or help desk for support
- Allows a password (that is changed in the cloud) to be written back to on-premises
- Convenient cloud-based way for your users to reset their on-premises passwords wherever they are.
- This is required for the Self-service password reset feature (SSPR and password change)
- Optional feature of Azure AD Connect (Part of Azure AD Premium edition)

Security

- All communication takes place over SSL
- Registration of public/private key pairs for transport and encryption, you keep the private keys

Works with Password Write back for Hybrid Identities

Chapter

1.2.3

Device Registration in Azure AD

🎯Identify the components needed for hybridization

Why should you bring devices in Azure AD?

Access control and identity protection

- Conditional access based on device policies
- Azure AD Identity Protection through advanced AI

Ease of deployment and management

- Autopilot, bulk provision & self-service deployment
- Modern device management using Intune
- Manage device identities in Azure AD portal

Seamless, secure and productive experiences for your users

- Single Sign On (SSO) to cloud and on-premises apps/resources
- Password-less credentials for secure & easy sign in
- Settings roamed across devices

What is Device Registration?

Giving to the device an Azure Identity:

- Device registration in Azure AD is the foundation for device-based conditional access scenarios.
- Can then be used to enforce conditional access policies for applications.
- Allows you to create additional conditional access rules that enforce access from devices to meet your standards for security and compliance by using compatible MDM

Azure Device Registration benefits

End users benefits:

- Enables SSO to Office365, SaaS Apps & Services from anywhere
  - By providing **Primary Refresh Token** PRT
- Enjoy roaming of OS settings across joined devices with Windows 10/11
- Be able to access the Windows Store for Business
- Have the convenience of Windows Hello
- Seamless MFA with Windows 10/11
- Seamless SSO authN iOS/Android/Windows

On the other hand, organizations:

- Can have control over these devices
- Make sure that users are accessing your resources from devices that meet standards for security and compliance
- Reduce the risk of credential theft by implementing Microsoft Hello

Device used for work

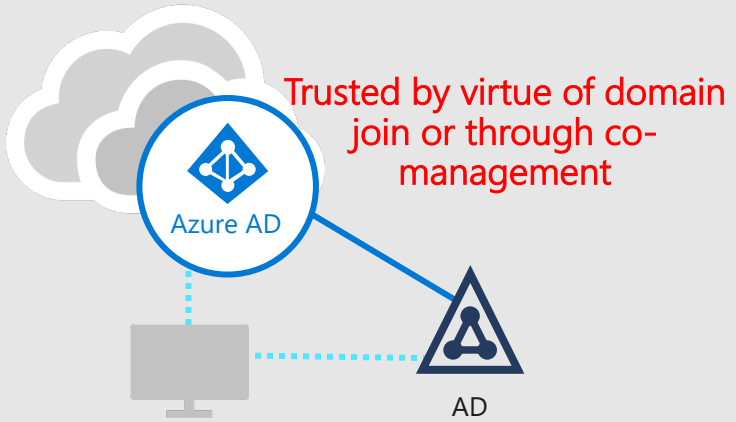
Corporate Owned Devices (COD)

- AD joined machine (Windows PCs)
- Mobile Devices (PCs/Laptops, Macs, Tablets, Phones)

Bring Your Own Device (BYOD)

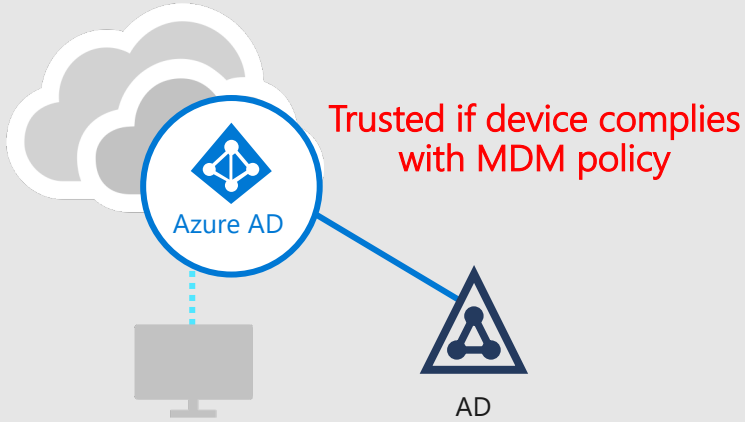
- Workstation (Windows PCs, Macs)
- Mobile Devices (PCs/Laptops, Macs, Tablets, Phones)

Summary view of devices in Azure AD



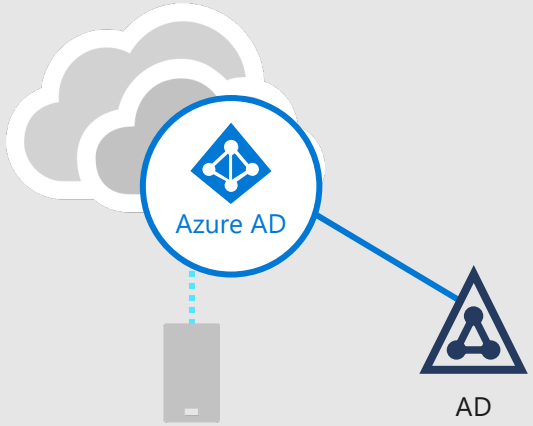
Hybrid Azure AD joined

- Corporate owned (COD)
- Joined to both on-premises AD and Azure AD
- Applicable to Windows current and down-level client/server
- Join happens automatically, controlled by admin config
- Managed by GPO/SCCM. CO-Management for cloud value



Azure AD joined

- Corporate owned (COD)
- Joined to Azure AD only (but access to on-premises resources)
- Windows 11, 10
- Join requires user interaction, controlled by Azure AD configuration
- Managed by MDM. Co-management for interim gaps
- SSO available to on-premises resources



Azure AD registered

- Typically, user owned device (BYOD)
- But can be corporate owned (COD)
- IOS, Android, MacOS & Windows
- Registered / known to Azure AD
- Can be managed by MDM
- Managed by MDM. Co-management for interim gaps

Adv.

Comparison of Device Join Scenarios

	Hybrid Azure AD Join	Azure AD Join	Azure AD Registered
Definition	Joined to on-premises AD and Azure AD requiring organizational account to login to the device	Joined only to Azure AD requiring organizational account to login to the device	Registered to Azure AD without requiring organizational account to login to the device
Primary Audience	Suitable for hybrid organizations with existing on-premises AD infrastructure Applicable to all users in an organization	Suitable for both cloud-only and hybrid organizations. Applicable to all users in an organization	Applicable to all users with the following criteria: • Bring your own device (BYOD) • Corporate owned mobile devices
Device ownership	Organization	Organization	User or Organization
Device types	Windows 11, 10, 8.1 and 7 Windows Server 2008/R2, 2012/R2, 2016/2019	All Windows 11, 10 devices	Windows 11, 10, **iOS, Android and MacOS**
Provisioning	Windows 11, 10, Windows Server 2016/2019 • Domain join by IT and auto-join via Azure AD Connect or Azure AD Federation Services (ADFS) config • Domain join by Windows Autopilot and auto-join via Azure AD Connect or ADFS config Windows 8.1, Windows Server 2012 R2/2012 • Requiring MSI	• Windows OOBE or Settings • Bulk enrollment • Windows Autopilot	• Windows 11, 10 – Settings or Office apps • iOS/Android – Company Portal or Microsoft Authenticator app • MacOS – Company Portal
Device login options	Organizational accounts using: • Password during OOBE & Win Logon • Windows Hello for Business during Win Logon • FIDO2.0 keys during Win Logon	Organizational accounts using: • Password during OOBE & Win Logon • Windows Hello for Business during Win Logon • FIDO2.0 keys during OOBE & Win Logon • Phone Signing during OOBE	End-user local credentials: • Password/Windows Hello for Windows 11, 10 • PIN, Biometrics or Pattern for others
Management	• Group Policy • Intune • SCCM standalone or co-management with Intune	• Intune • Co-management with Intune and SCCM • SCCM standalone	• Intune
Key capabilities	• **SSO to both cloud and on-premises resources** • Conditional Access through Domain join or through Intune if co-managed • Self-Service Password Reset on lock screen • Enterprise State Roaming across devices	• **Single Sign On (SSO) to both cloud and on-premises resources** • Conditional Access when enrolled into MDM like Intune • Self-Service Password Reset on lock screen • Enterprise State Roaming across devices	• **SSO to cloud resources** • Conditional Access **when enrolled into Intune**

Adv.

What happen next when device is register to Azure AD?

The device is registered with Azure AD

NAME	ENABLED	OS	VERSION	JOIN TYPE	OWNER
WS-WIN10-AADJ-L	Yes	Windows	10.0.17763.316	Azure AD joined	Tim

NAME	ENABLED	OS	VERSION
GoogleAndroid SDK built ...	Yes	Android	8.0.0

NAME	ENABLED	OS	VERSION	JOIN TYPE	OWNER
WS-WIN10-DRS-L	Yes	Windows	10.0.17763.253	Azure AD registered	Tim

The device has been populated with some attribute:

- Device Trust type:
- Device State …

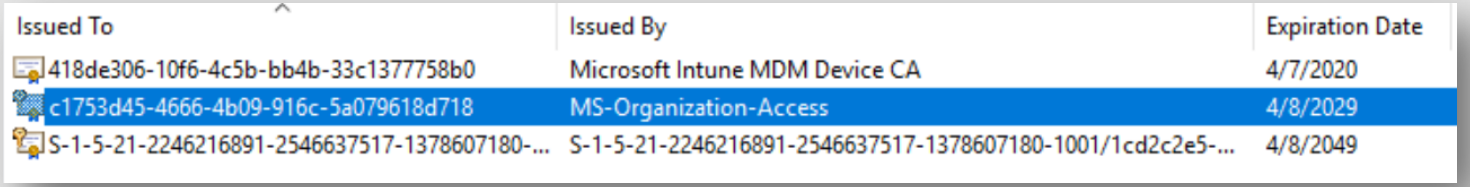
```
DeletionTimestamp                 :
ObjectId                          : 6ebc9bdf-4300-429b-8470-b747e9534427
ObjectType                        : Device
AccountEnabled                    : True
AlternativeSecurityIds            : {class AlternativeSecurityId {
                                      IdentityProvider:
                                      Key: System.Byte[]
                                      Type: 2
                                    }
                                    }
ApproximateLastLogonTimeStamp     : 08/04/2019 19:25:52
ComplianceExpiryTime              :
DeviceId                          : c1753d45-4666-4b09-916c-5a079618d718
DeviceMetadata                    :
DeviceObjectVersion               : 2
DeviceOSType                      : Windows
DeviceOSVersion                   : 10.0.17763.253
DevicePhysicalIds                 : {[USER-GID]:40bf68c1-208f-4e43-a4b3-49834dfcff7b:6966503360191218,
                                    [GID]:g:6966503360191218,
                                    [USER-HWID]:40bf68c1-208f-4e43-a4b3-49834dfcff7b:6755409916764607,
                                    [HWID]:h:6755409916764607}
DeviceTrustType                   : Workplace
DirSyncEnabled                    :
DisplayName                       : WS-WIN10-DRS-L
IsCompliant                       : True
IsManaged                         : True
LastDirSyncTime                   :
ProfileType                       : RegisteredDevice
SystemLabels                      : {}
```

Adv.

What happen next when BYOD device is register to Azure AD?

A certificate is placed on the device

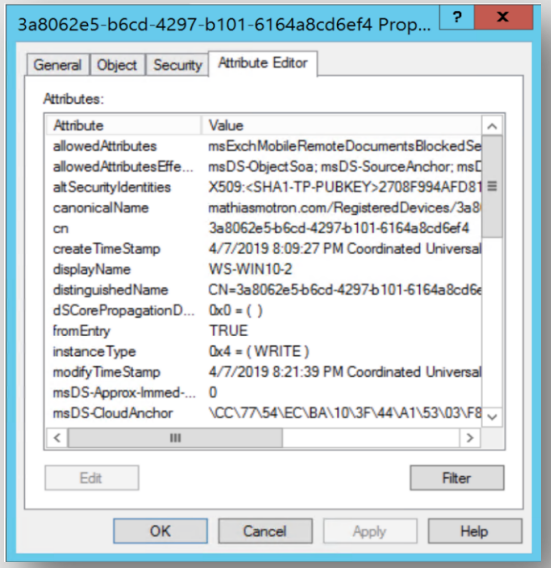
= The device has an identity



Windows 10 generates Private/Public keys for Windows Hello, and register public key to Azure AD (if applicable)

```
+----------------------------------------------------------------------+
| User State                                                           |
+----------------------------------------------------------------------+

                NgcSet : YES
              NgcKeyId : {6D27D1E5-6A92-43F4-B0D3-C86B3AA99A6B}
              CanReset : NO
       WorkplaceJoined : YES
      WorkAccountCount : 1
         WamDefaultSet : NO
```



If Device Writeback is enable, object is writeback on OnPrem AD

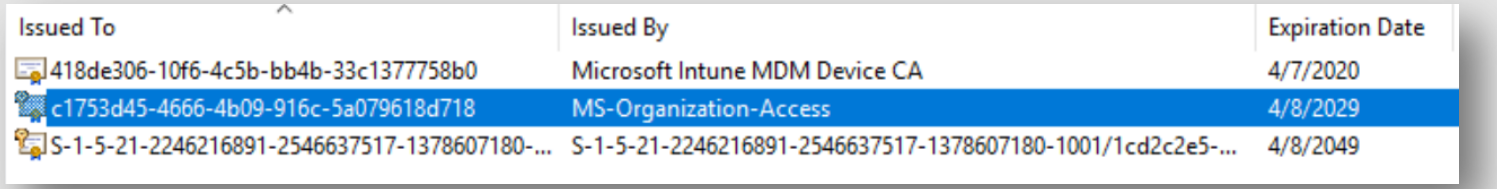
for OnPrem Conditionnal Access with ADFS

→ Optionally enrolled in the MDM system that your organization has configured

Adv.

What happen next when Windows is Azure AD Join?

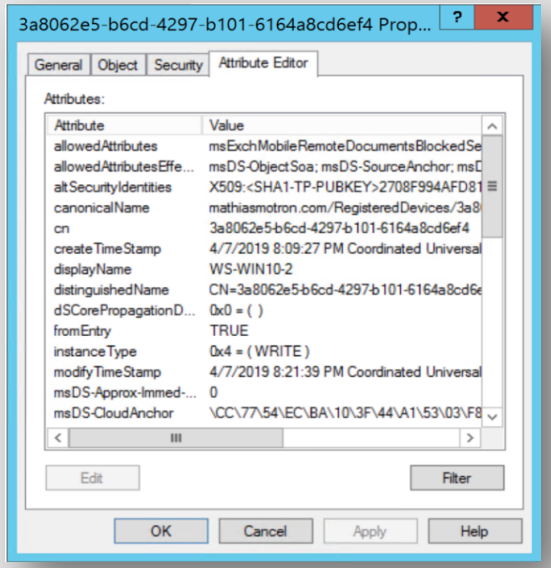
A certificate is placed on the device = The device has an identity



Windows generates Private/Public keys for Windows Hello, and register public key to Azure AD (if applicable)

If Device Writeback is enable, object is writeback on OnPrem AD

Automatic MDM enrollment (enabled by default)



Adv.

What happen next when Windows is Hybrid Azure AD Joined?

A certificate is placed on the device = The device has an identity

- In Local Computer Store for Automatic Registration
- In User Store for Manual Registration

If Device Writeback is enable, object is writeback on OnPrem AD for OnPrem Conditionnal Access with ADFS and OnPrem SSO

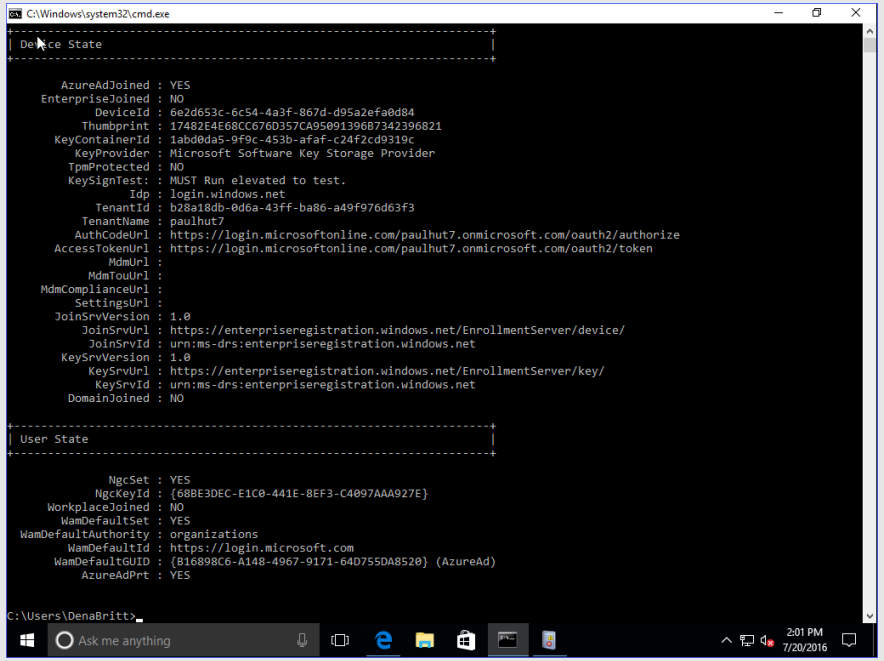
Every Logon/Unlock will make the device obtain:

- An SSO token (Kerberos TGT) from AD
- An SSO token (PRT) from Azure
- An SSO token (PRT) from AD FS 2016 if domain is federated
- SSO Anywhere (Windows 10)

Adv.

Get Azure AD Joined status

Dsregcmd.exe /status is a great in-box tool that can be used to provide information about device and user state.



In this example, we can see that we are Azure AD Joined and our Passport (called NGC in the tool) is configured

Other interesting bits are our Azure AD Device ID, device thumbprint, and Azure AD Tenant ID which we discovered earlier in our event logs and registry

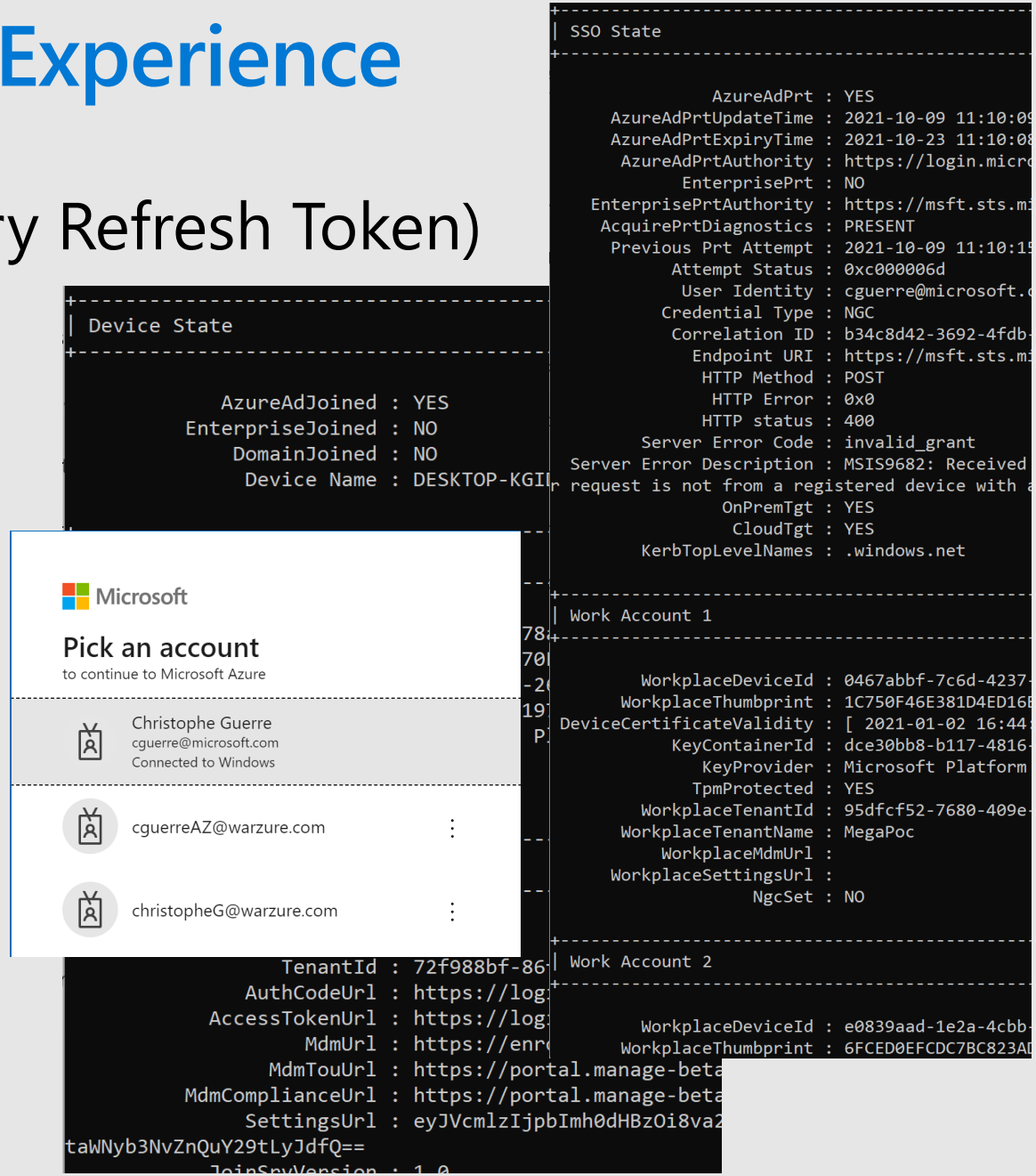
What is a PRT?

- Primary Refresh Token (PRT): is a key artifact of Azure AD authentication on Windows 10 or newer, Windows Server 2016 and later versions, iOS, and Android devices.
  - It is a JSON Web Token (JWT) specially issued to Microsoft first party token brokers
- Enable single sign-on (SSO) across the applications used on those devices
- Device registration is a prerequisite for device-based authentication in Azure AD.
  - A PRT is issued to users only on registered devices
- Once issued, a PRT is valid for 14 days and is continuously renewed as long as the user actively uses the device

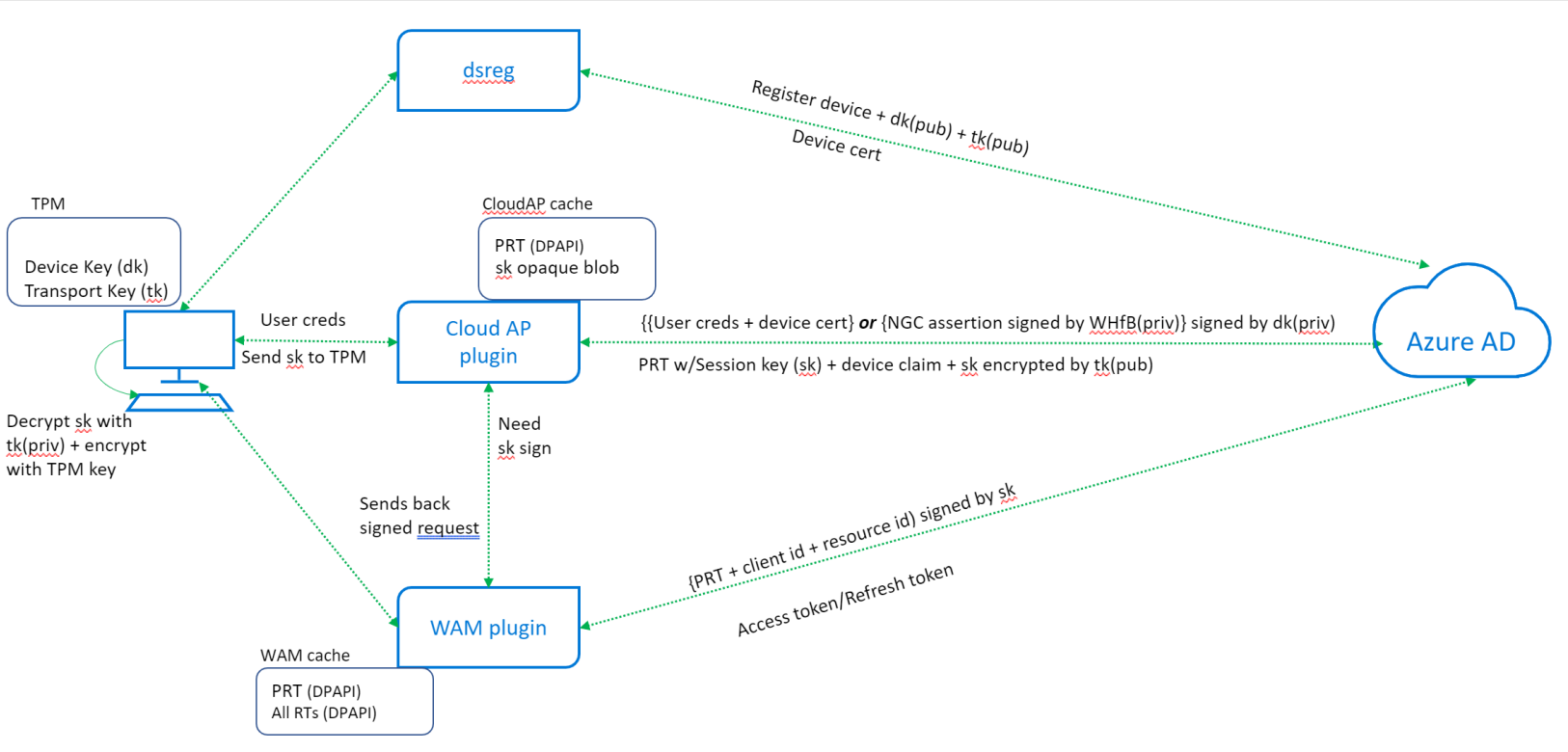
Primary Refresh Token User Experience

SSO experience through PRT (Primary Refresh Token) across the applications

- Azure AD Join
  - (and Hybrid Azure AD Join)
  - Check with Dsregcmd /status
- Every credential used to unlock has its own PRT
- When PRT is involved, you will see fewer prompts than you'd expect
  - No prompt = worsened security is a misconception/FUD
- PRT refreshes and device unlock are considered a prompt – proof of presence in NIST



PRT in action & Token Binding



Primary Refresh Token:

- Requested by Cloud AP
- Binding Key protected by the TPM (2.0)
- Refreshes every device unlock & WAM request
  - But not more often than 4h

Token Binding:

- Protect token from replay attack
- 2 approaches:
  - TLS TB
  - PoP (Proof of Possession)

[Primary Refresh Token (PRT) and Azure AD - Azure Active Directory | Microsoft Docs]

Chapter

1.2.4

AD user authentication in Azure AD

🎯 Describe the authentication mechanisms

For Reminder Azure AD Authentication

We call **hybrid identity**

A pan on-premises and cloud-based capabilities create a common user identity for authentication and authorization to all resources, regardless of location.

To achieve hybrid identity with Azure AD, one of these three authentication methods can be used:

PHS

PTA

- Managed (**P**assword **H**ash **S**ync)
  - The Hash (Hash (Password)) is synchronized from AD but it's AAD who authenticate the account
  - The simpliest
- **P**ass-**T**hrough **A**uthentication
  - Agent relay authentication from AAD to AD. It's AD who authenticate the user
- Federation
  - AAD is federated to AD. It's AD who authenticate the user
  - The Hash (Hash (Password)) (PHS) can be synchronized from AD but it's not mandatory. Recommended for DRP

Hybrid Authentication – Managed (PHS) scenario

It’s a Cloud authentication -> it’s Azure AD that authenticate

PHS is the simplest way to enable authentication for on-premises directory objects in Azure AD

- Users can use the same username and password that they use on-premises

Require password hash synchronization

- Passwords are never stored in clear text or encrypted with a reversible algorithm in Azure AD
- It’s Hashes of user passwords salted 1000x

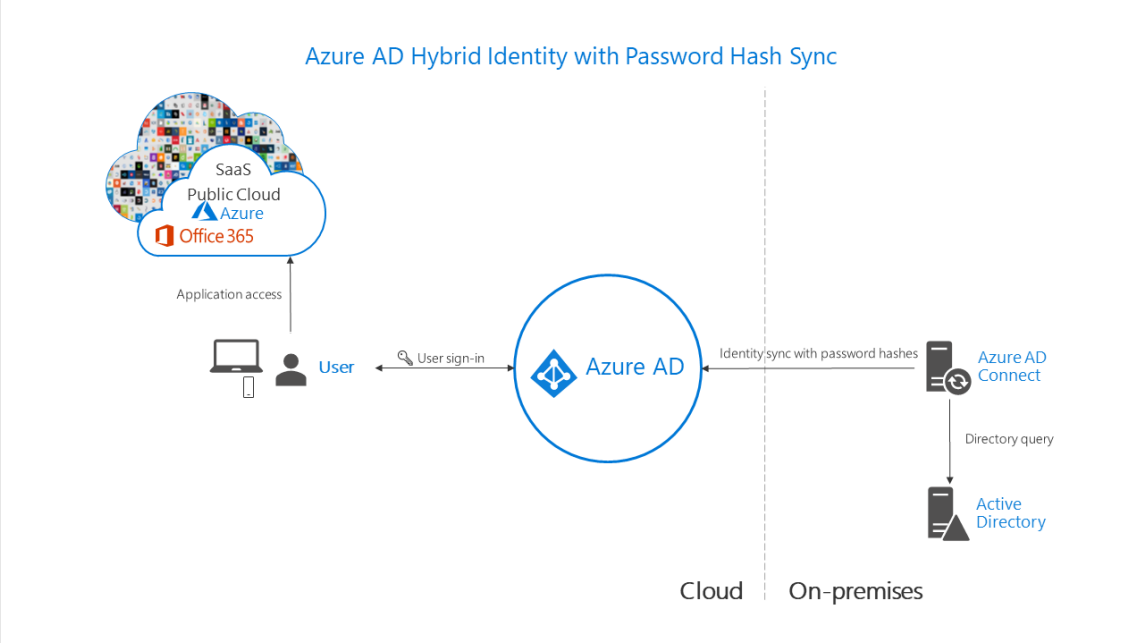
Password Hash Synchronization

Hashes of user passwords are synchronized from on-premises Active Directory to Azure AD.

Password sync schedule runs every two minutes. It's not possible change it.

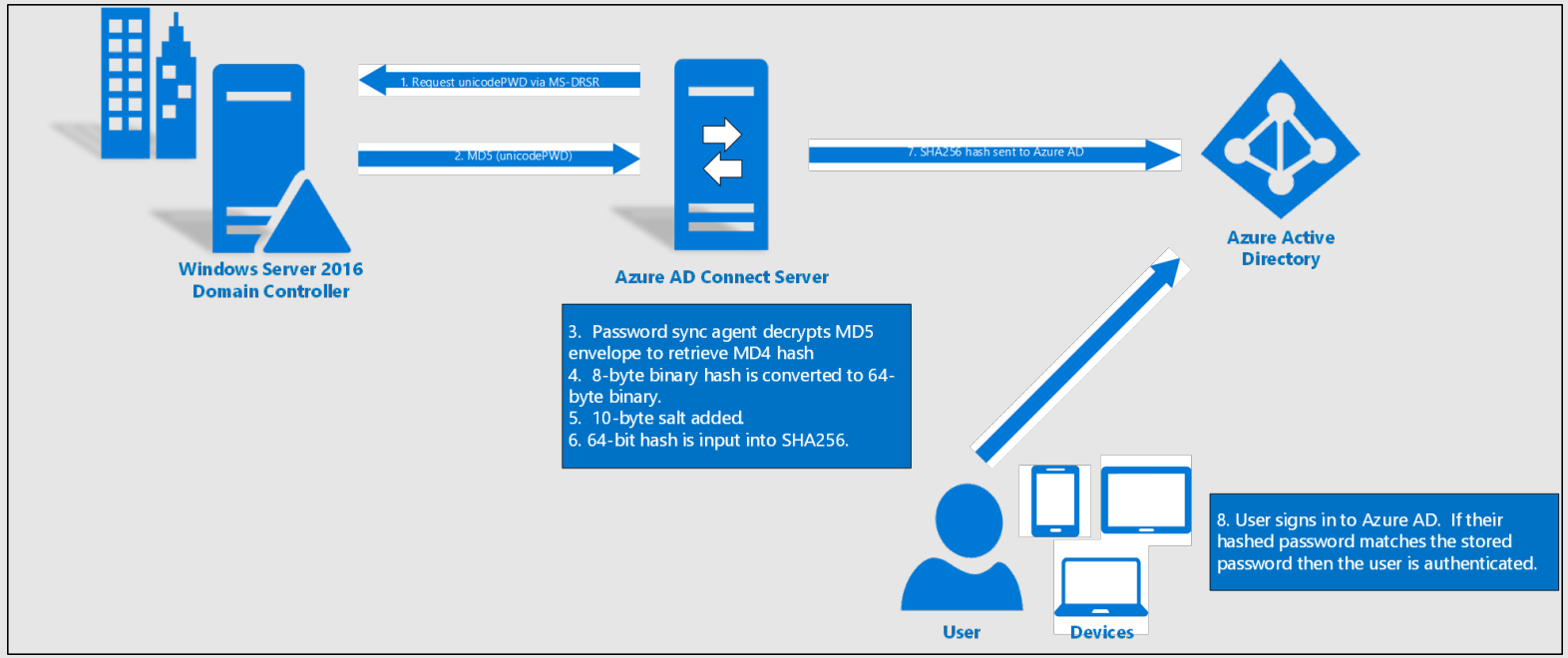
The passwords are never sent to Azure AD or stored in Azure AD in clear text.

The original MD4 hash is not transmitted to Azure AD. The hash from Azure AD cannot be used in an on-premises pass-the-hash attack.



Adv.

Password Hash Synchronization



1. Every two minutes, the password hash synchronization agent on the AD Connect server requests stored password hashes (the unicodePwd attribute) from a DC.
2. Before sending, the DC encrypts the MD4 password hash
3. AD Connect decrypts the received data back to its original MD4 format.
4. The password hash synchronization agent expands the 16-byte binary password hash to 64 bytes
5. The password hash synchronization agent adds a per user salt
6. The password hash synchronization agent then combines the MD4 hash plus the per user salt, 1000 iterations of the HMAC-SHA256 keyed hashing algorithm are used.
7. The PHS agent then transmits the string from to Azure AD over SSL.
8. When a user attempts to sign in, the password is run through the same process.

Adv.

Security Considerations

The **plain-text version** of your password is **not exposed** to the password synchronization feature, to Azure AD, or any of the associated services.

User **authentication takes place against Azure AD** rather than against the organization's own Active Directory instance.

If any concerns about password data in leaving the on-premises, consider the fact that the SHA256 password data stored in Azure AD as hash of the original MD4 hash is significantly more secure than what is stored in Active Directory.

Further, because this SHA256 hash cannot be decrypted, it cannot be brought back to the organization's Active Directory environment and presented as a valid user password in a pass-the-hash attack.

Adv.

Password policy considerations

Password complexity policy	Password expiration policy
Password complexity policies on-premises override complexity policies in the cloud for synchronized users.	If a user is in the scope of password synchronization, the cloud account password is set to Never Expire
Passwords for users in the cloud are still subject to password policies as defined in the cloud.	You can continue to sign-in to your cloud services by using a synchronized password that is expired for the on-premises password policies*
	The cloud password is updated the next time you change the password in the on-premises environment.
	An administrator can manually reset your password by using Windows PowerShell
	AccountExpires attribute is not synchronized to Azure AD

** you can force users to comply with your Azure AD password expiration policy by enabling the EnforceCloudPasswordPolicyForPasswordSyncedUsers feature*

Pass-through Authentication (PTA) Scenario

Pass-through Authentication allows your users to sign-in to both on-premises and cloud-based applications using the same passwords.

This feature is an alternative to Azure AD Password Hash Synchronization

You can combine Pass-through Authentication with the Seamless Single Sign-On feature

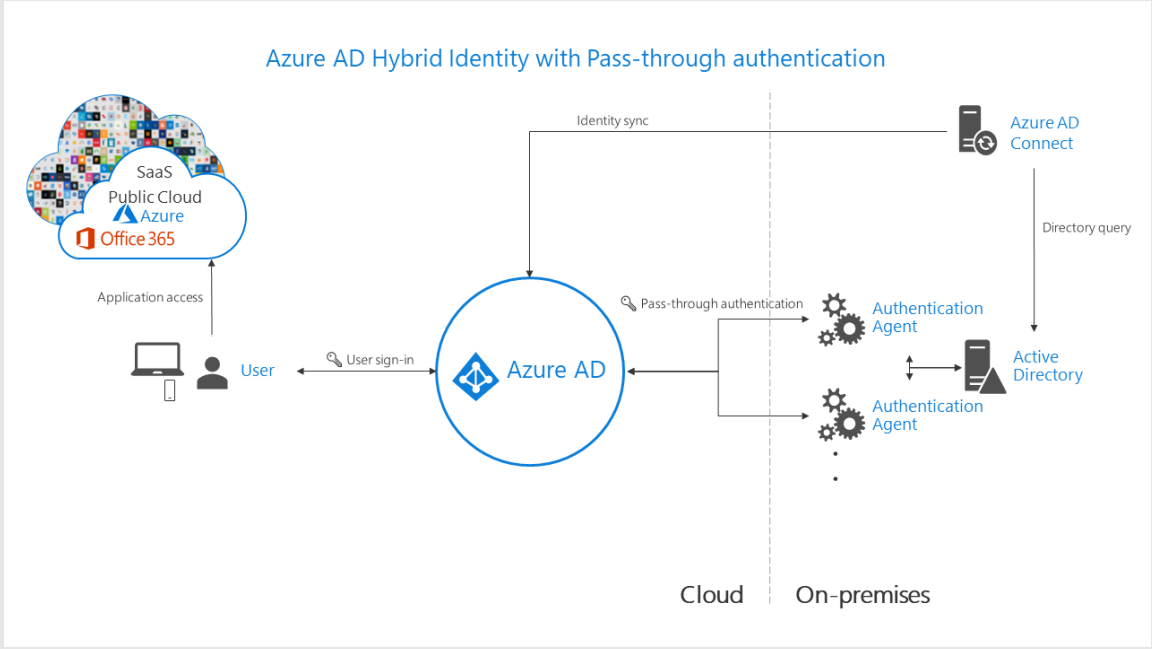
Feature highlights

Supports sign-in on web browser-based applications and into Microsoft Office client that use modern authentication.

Sign-in usernames can be either the on-premises default username (userPrincipalName) or another attribute configured in Azure AD Connect (known as Alternate ID).

The feature works seamlessly with conditional access features such as Multi-Factor Authentication (MFA) to help secure your users.

Integrated with cloud-based self-service password management, including password writeback to on-premises Active Directory and password protection by banning commonly used passwords.



Adv.

Feature highlights

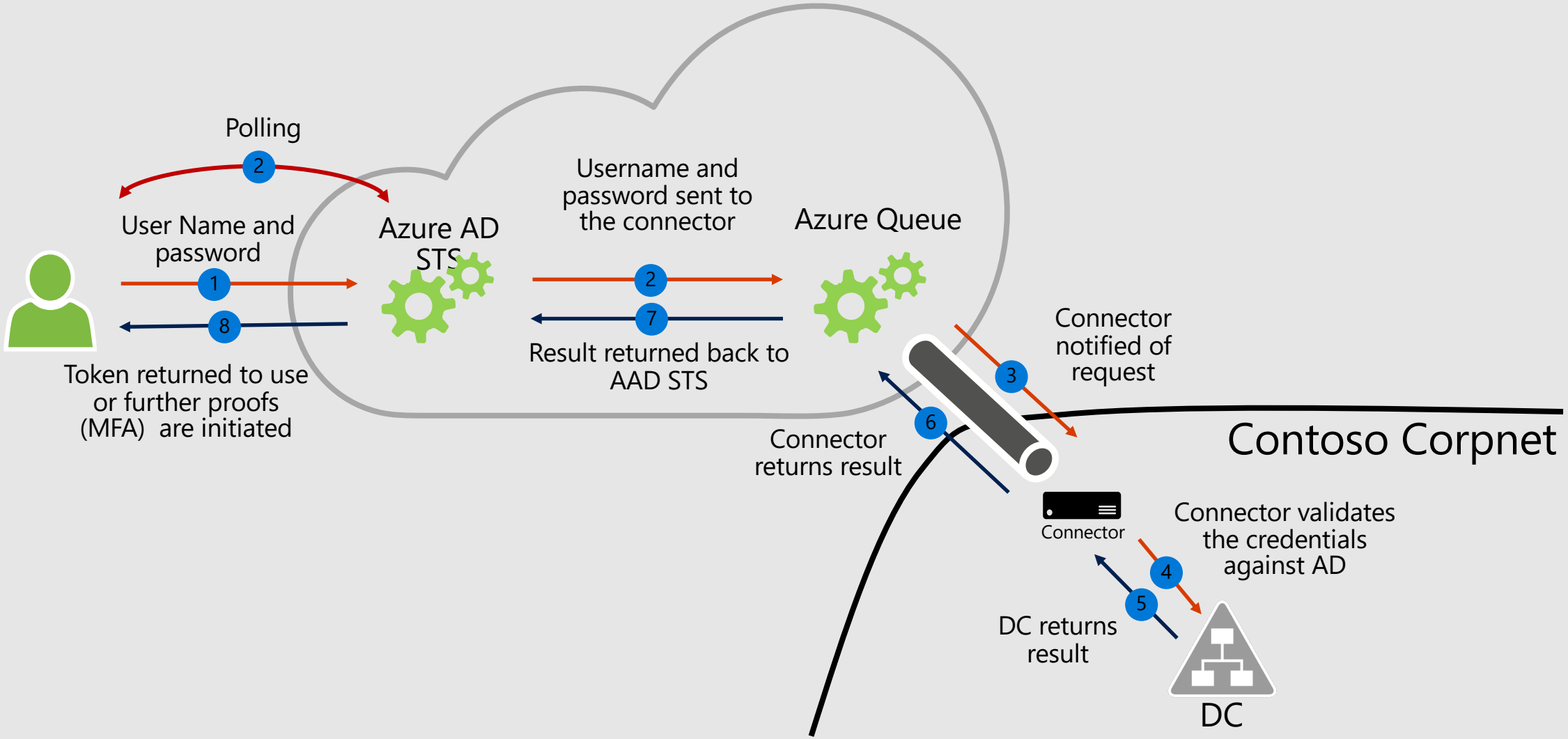
Multi-forest environments are supported if there are forest trusts between your AD forests and if name suffix routing is correctly configured.

It uses a lightweight on-premises agent that listens for and responds to password validation requests.

Installing multiple agents provides high availability of sign-in requests.

It protects your on-premises accounts against brute force password attacks in the cloud.

How does it work



Adv.

Supported scenarios

User sign-ins into all web browser-based applications.

User sign-ins into Office 365 client applications that support modern authentication.

Azure AD Join for Windows 10 devices.

Exchange ActiveSync support.

Adv.

Unsupported scenarios

Detection of users with leaked credentials .

Azure AD Domain Services needs Password Hash Synchronization to be enabled on the tenant .

- Therefore, tenants that use Pass-through Authentication only don't work for scenarios that need Azure AD Domain Services.

Pass-through Authentication is not integrated with Azure AD Connect Health .

Adv.

Smart Lockout

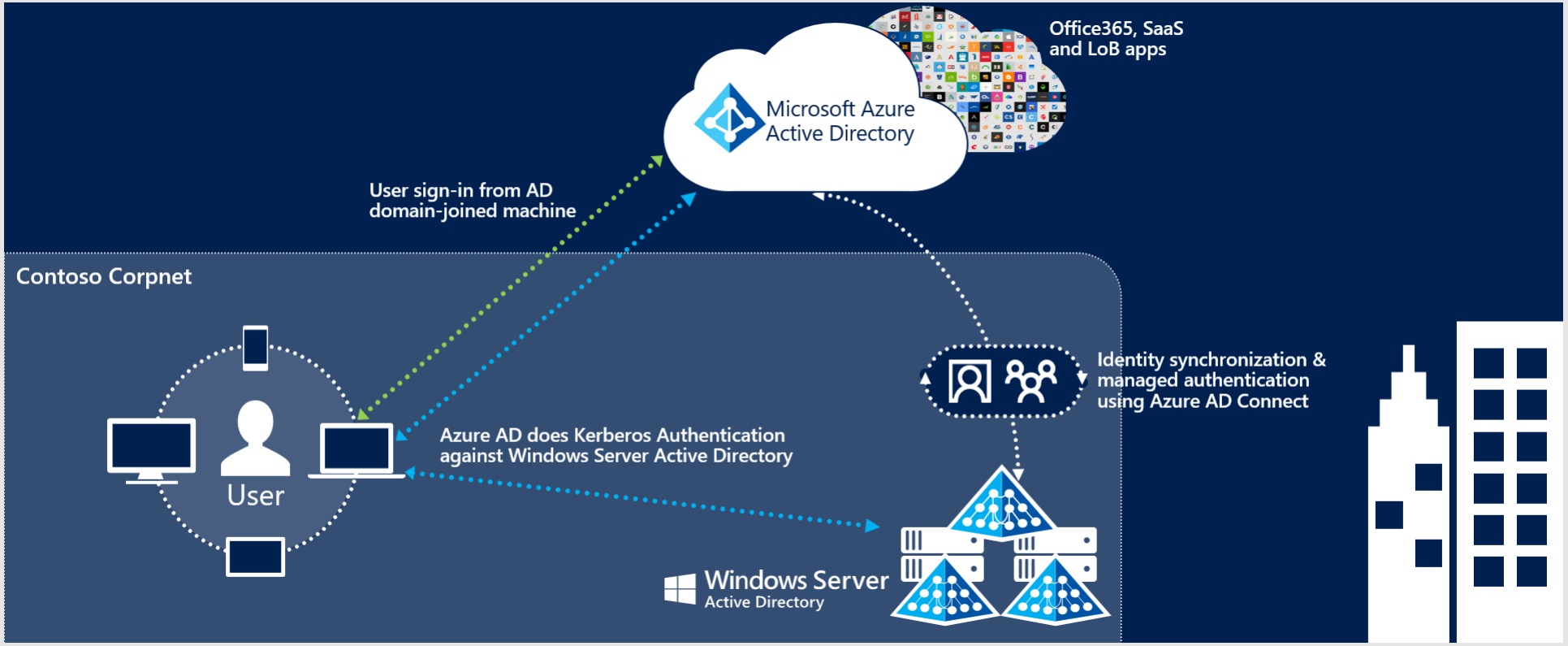
Smart Lockout works by keeping track of failed sign-in attempts, and after a certain Lockout Threshold, starting a Lockout Duration.

Because Pass-through Authentication forwards password validation requests onto your on-premises Active Directory (AD), you need to prevent attackers from locking out your users' AD accounts.

The default Lockout Threshold is 10 failed attempts, and the default Lockout Duration is 60 seconds.

Seamless SSO

Seamless SSO can be combined with either the Password Hash Synchronization or Pass-through Authentication sign-in methods.



Overview

It is supported on web browser-based clients and Office clients that support modern authentication on platforms and browsers capable of Kerberos authentication

OS\Browser	Internet Explorer	Microsoft Edge	Google Chrome	Mozilla Firefox	Safari
Windows 10	Yes*	No	Yes	Yes***	N/A
Windows 8.1	Yes*	N/A	Yes	Yes***	N/A
Windows 8	Yes*	N/A	Yes	Yes***	N/A
Windows 7	Yes*	N/A	Yes	Yes***	N/A
Windows Server 2012 R2 or above	Yes**	N/A	Yes	Yes***	N/A
Mac OS X	N/A	N/A	Yes***	Yes***	Yes***

*Requires Internet Explorer versions 10 or above

**Requires Internet Explorer versions 10 or above. Disable Enhanced Protected Mode

***Requires additional configuration

How does it work?

A computer account named AZUREADSSOACCT (which represents Azure AD) is created in your on-premises Active Directory (AD).

The computer account's Kerberos decryption key is shared securely with Azure AD.

In addition, two Kerberos service principal names (SPNs) are created to represent two URLs that are used during Azure AD sign-in.

We highly recommend that you roll over the Kerberos decryption key at least every 30 days.

The last but not the least - Federation Scenario

Require a strong infrastructure with at least:

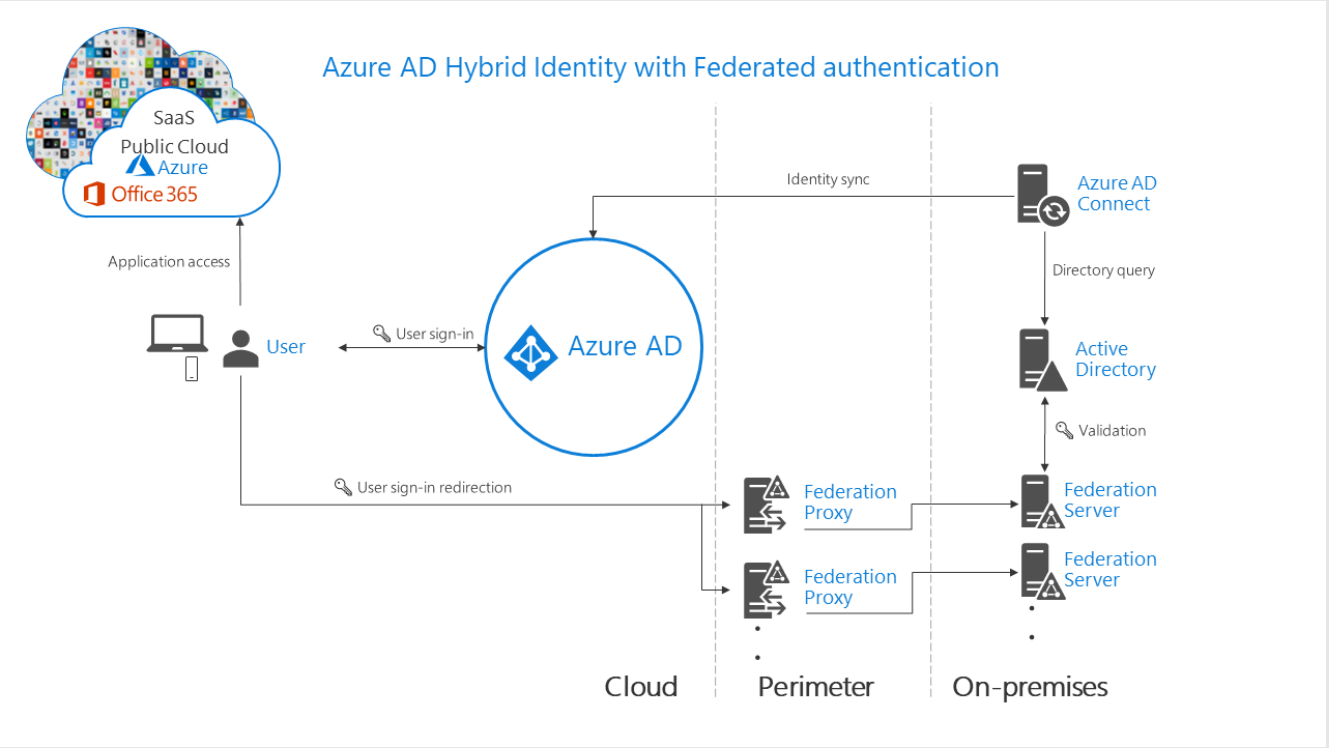
- 2 Federation Servers to provide redundancy
  - It's a SPOF for the authentication
- 2 Federation Proxy to provide security
  - Secure external authentication

Provide full SSO experience

Operation close to PTA

Allow advanced scenarios

- CBA
- Alternate LoginID



AD FS Federation Concepts

Federation is a process that enables distributed authentication, and authorization across organizational and platform boundaries.

Federation is based on a trust relationship that is protected by a trust key or password.

Federation provides single sign-on (SSO) feature to web applications and web services across security boundaries:

- Users need to remember only a single username and password.

Federation relieves the burden of extranet provisioning:

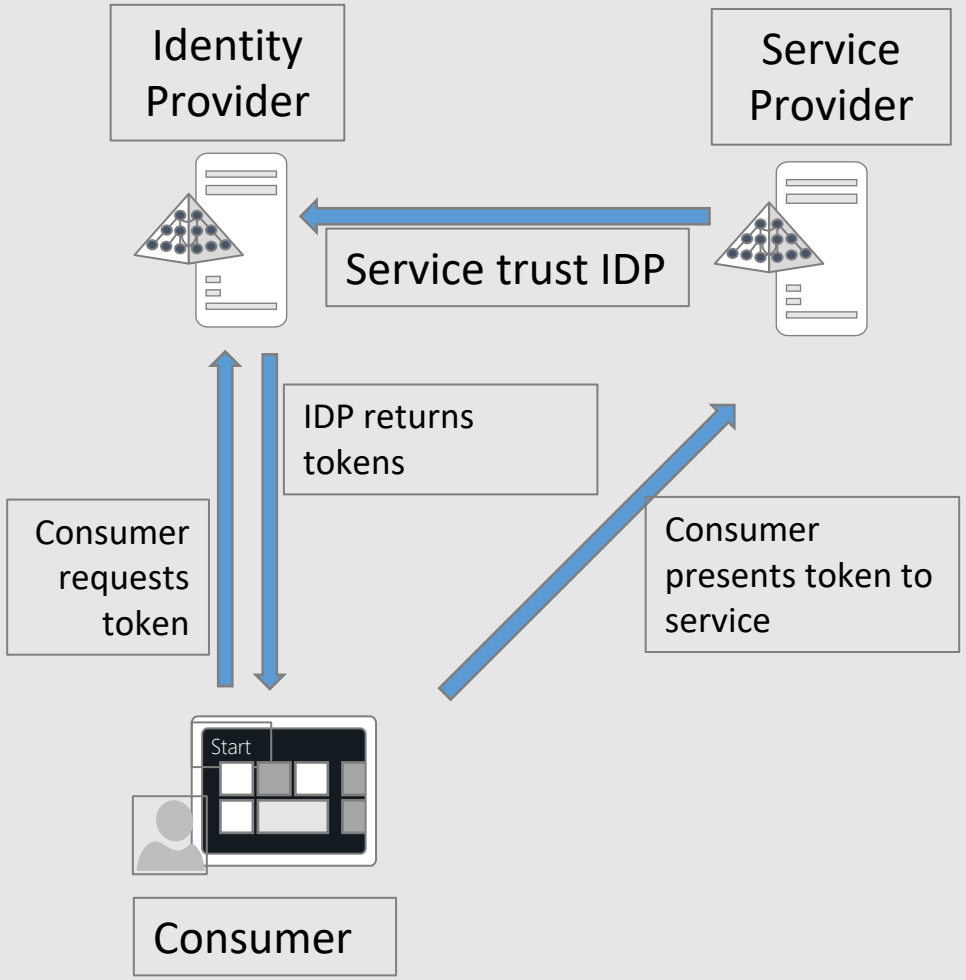
- Management of user accounts is minimized.

AD FS Federation Concepts: Components Involved

Different parties participate in Federation in different ways. The three options are:

- Identity Providers, which issue digital identities
- Relying Parties (also known as Service Providers), which require identities
- Identities, which are individuals and other entities about whom claims are made

Note: In many cases, the participants in a Federation play more than one role, and often all three roles.



AD FS Federation Concepts: Tokens

A token is a package of claims in a consistent format that can be consumed by an application (relying party)

Generated by an issuer – Security Token Service (STS):

- Issuer verifies the user identity
- Issuer gets claims from potentially multiple sources
- Issuer creates a token
- Issuer signs the token
- Token is optionally encrypted

Consumed by a relying party:

- Trusts one or more specific token issuers (STSs)
- Receives the token that was created by a trusted issuer
- Verifies the signature on the token
- Decrypts the token if it was encrypted
- Extracts the claims and uses them

AD FS Federation Concepts: Claims

A claim is a statement about one entity that is asserted by some other entity to be true. Some examples are:

- A user's universal principal name (UPN) is: john.smith@contoso.com
- The user is over 21 years of age
- The user is in the sales role

Accuracy of claims:

- You cannot always verify if the claim is true
- Given to you by a trusted source

Many different uses:

- Making application authorization decisions
- Application customizations (that is, "Welcome, John!")

ADFS Components

Federation Server

Holds configuration for trust relationships

Issues authentication tokens

Audits authentication attempts

Interfaces with other services (e.g. Device Registration Service)

Is Domain Joined

Configuration DB

- Stores AD FS Farm Configuration (trusts, certificates, logon page formatting etc.)
- Stores the configuration for the Web Application Proxy
- May utilize Windows Internal Database (WID) or SQL Server

Web Application Proxy

Provide reverse proxy functionality for web applications

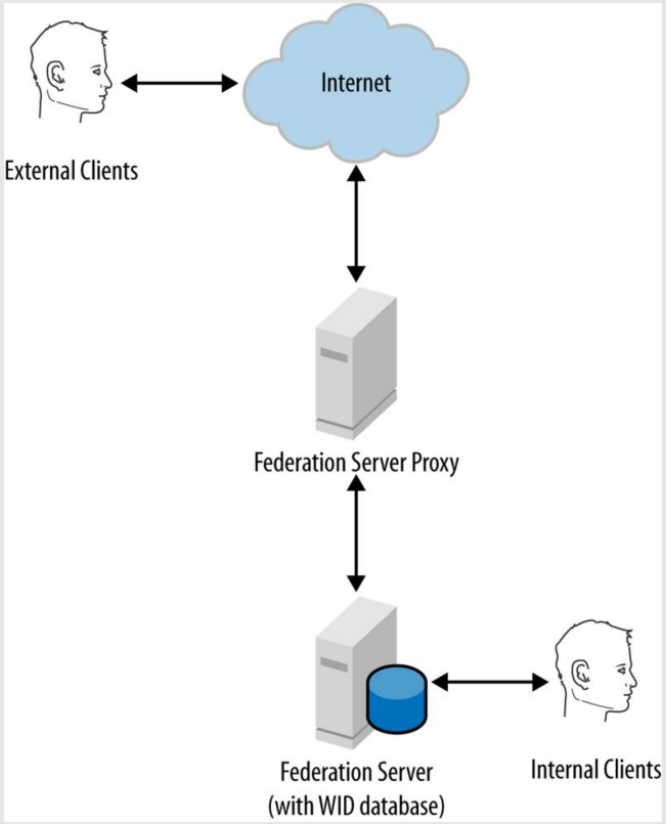
Proxy authentication requests to AD FS farm servers

Usually reside in a workgroup (not domain-joined)

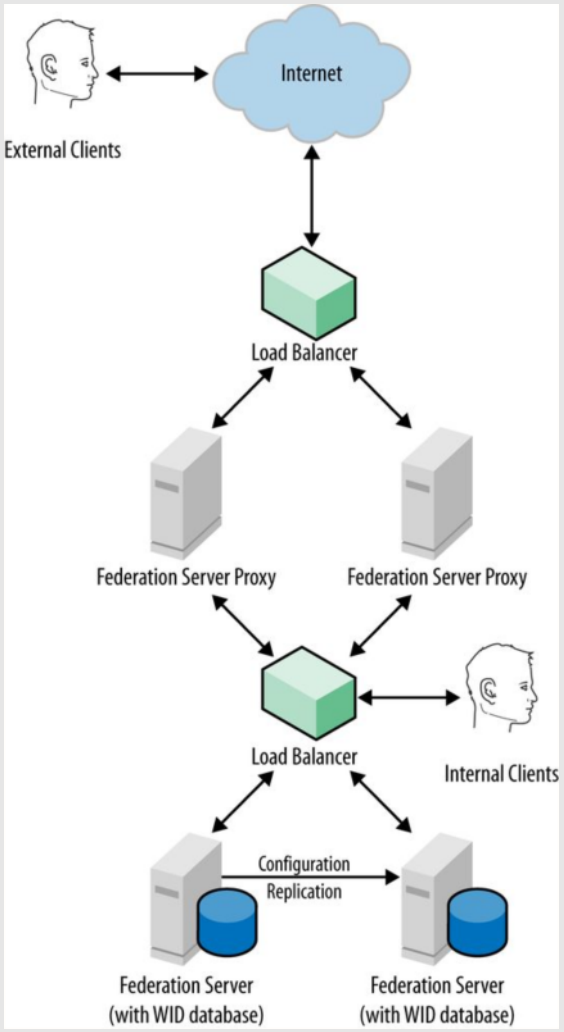
Tag authentication requests as coming from outside the LAN

Can be domain-joined for usage of Constrained delegation

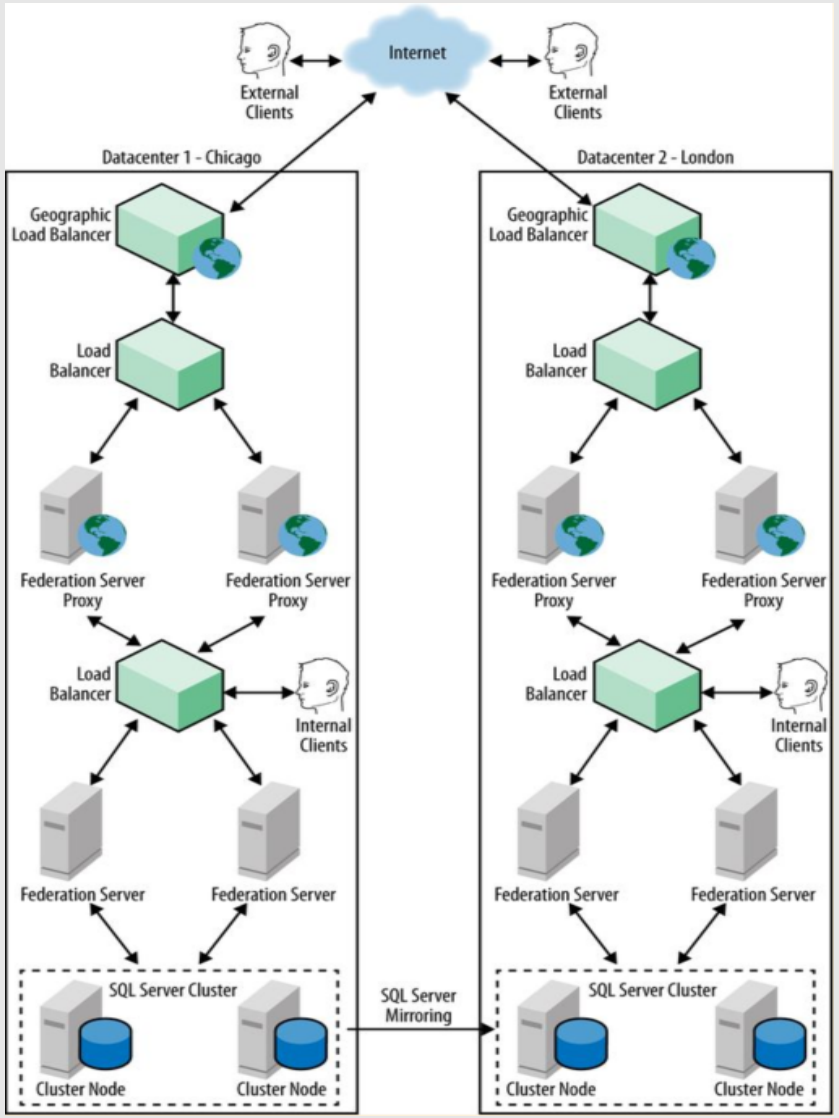
AD FS Topologies



Single Federation Server



Load-balanced ADFS Servers



Geographically Redundant ADFS Servers

AD FS Requirements: Certificates

Service communication Secure Sockets Layer (SSL) certificate:

- Is used to establish SSL connections
- Should be a third-party certificate
- The subject name should match the AD FS service name

Token signing certificate:

- Is used to sign SAML tokens sent to the applications
- Is created as a self-signed certificate by default
- It can be difficult to change/update, as all relying party trusts must be updated to trust a new certificate

AD FS Requirements: Certificates

Token encrypting/decrypting certificate:

- Is used to encrypt SAML tokens sent from claims providers to AD FS
- Is created as a self-signed certificate by default
- Can be difficult to change/update, as all claim provider trusts using token encryption must be updated to use new certificate

Adv.

AD FS Requirement: Domain Name System

Federation service name:

- A fully qualified domain name (FQDN) for the AD FS deployment
- Specified during the initial AD FS configuration
  - Example: sts.contoso.com
- The same federation service name is used internally and externally, which will require split-brain domain name system (DNS) in most scenarios.

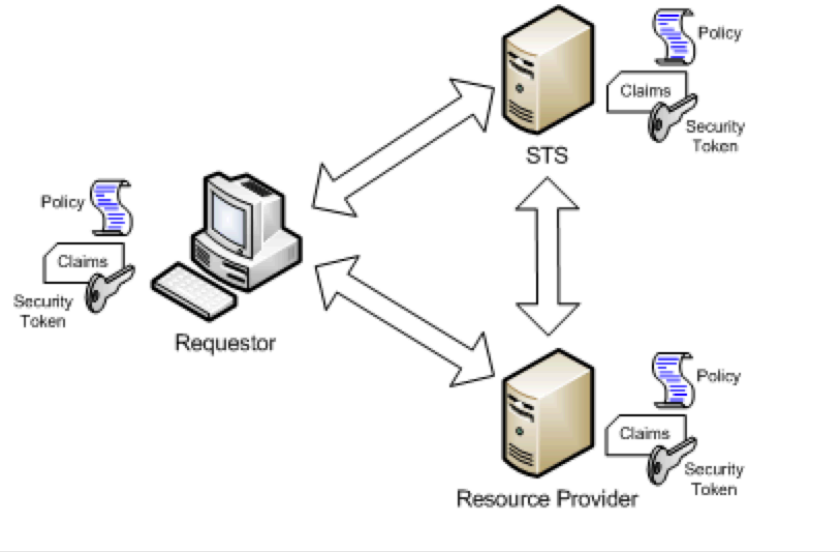
Use an A (host) record for the federation service name:

- Do not use a CNAME (alias), if you do Kerberos won't work

Authentication flow

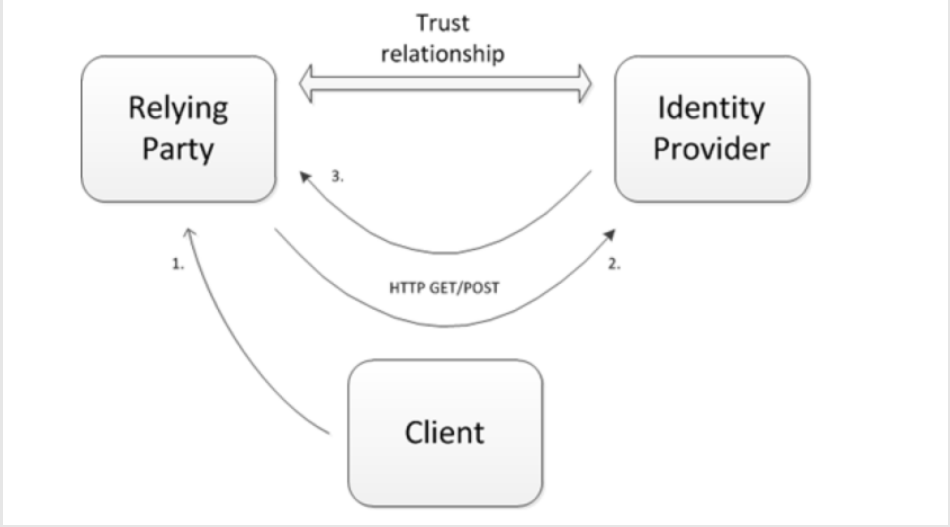
Active profile

Active authentication uses direct connection to request a token and login. In this case the protocol that is used is WS-Trust.



Passive profile

Passive authentication uses the browser to do redirects to the identity provider to request a token. The protocol that is used is WS-Federation



PTA vs ADFS – When use one or other?

	PTA	ADFS
Hardware	at least 2 server LB no required	2 ADFS + 2 WAP LB required
Firewall	No inbound port required	TCP 443 inbound
3rd SSL Certificate	Not required	Required
Password validation	On-premises	On-premises
"Where does the user enter their credentials?"	Cloud (encrypted) and send to on-premises	On-premises
Deploy / Maintenance	Low	Medium or High
Modern Authentication	Required	Not required

Some scenarios are not supported by PTA like Office365 access without Modern authentication support (ex: Office 2010)

ADFS gives you more flexibility if you need custom claims for your apps, SAML 1.1 token support, 3rd party MFA support, flexible protocol parameters, etc.

The Challenge of AlternateID

A User Principal Name (UPN) is required for authentication to Microsoft Azure AD

Azure AD requires all user login IDs to be fully internet routable

On-premises UPN uses a non-routable domain (ex. Contoso.local)

On-premises UPN cannot be changed due to local application dependencies

Important

Using Alternate ID in hybrid environments with Exchange and/or Skype for Business is **supported but not recommended**. Using the same set of credentials (e.g. the UPN) for on-premises and online provides the best user experience in a hybrid environment.

Microsoft recommends customers change their UPNs if possible, to avoid the need for Alternate ID.

Using alternate ID with Lync or Skype for business requires Lync Server 2013 or later. Customers who use Alternate ID should consider enabling Modern Authentication for Exchange in Office 365 for an improved user experience. In addition, customers using Skype for Business with mobile clients must ensure that the SIP address is identical to the user's mail address (and alternate ID).

Overview

Only available for username/password authentication

When WIA is performed UPN will be used for authentication

Requires global catalog to be reachable from the AD FS servers (Otherwise will fall back to UPN)

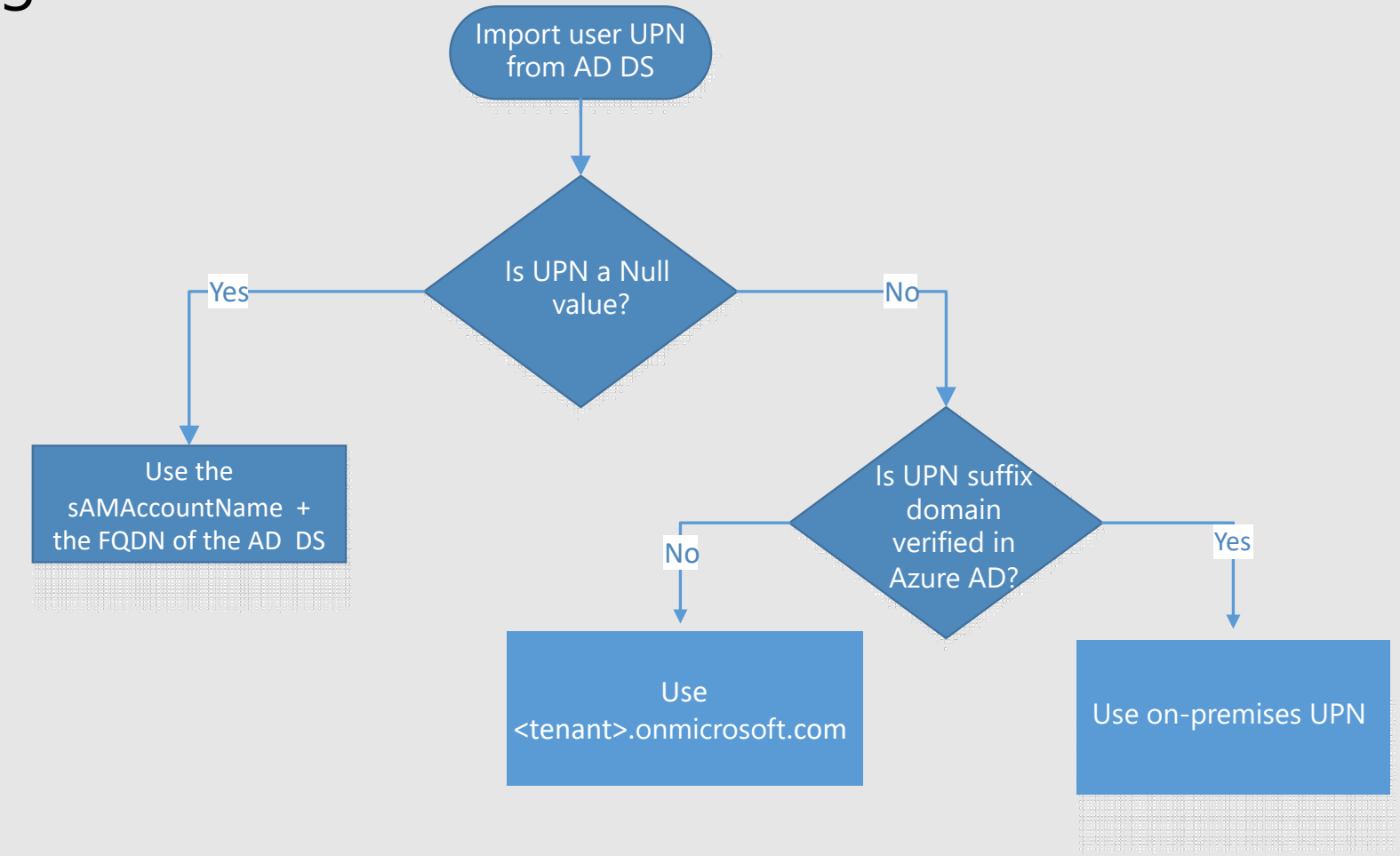
If the AD FS server finds duplicate user objects (ex: same e-mail address) across all the configured user account forests, it will fail the login

AD FS will try to authenticate with the ALID attribute first and then fall back to UPN if it cannot find an account identified by the ALID

AD FS adds additional performance counters for ALID

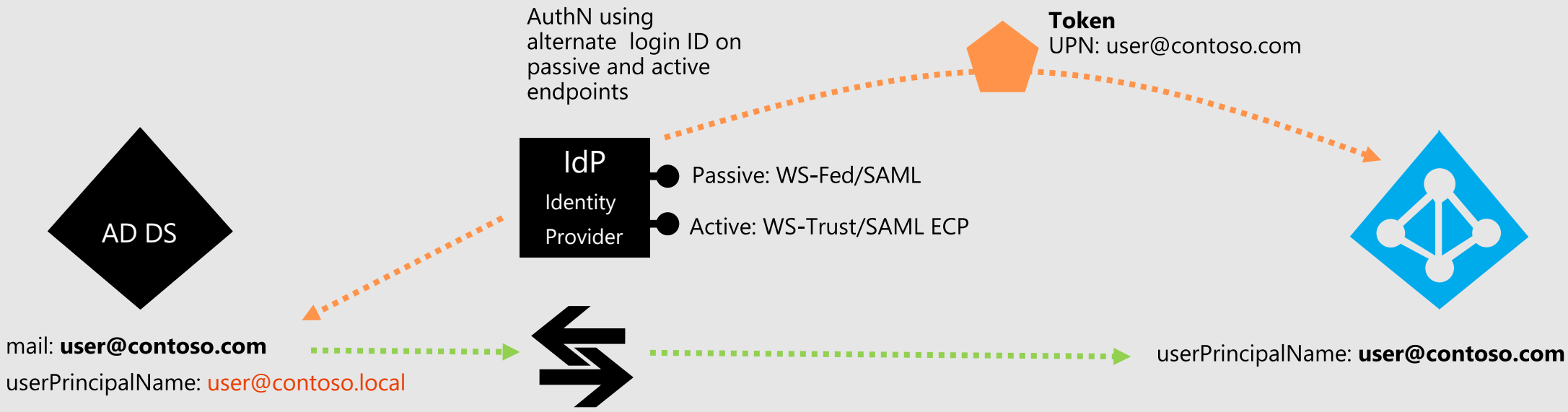
UPN Synchronization Process

Microsoft Azure AD Connect Tool (AAD Connect) defaults to the following logic for UPN creation in Azure AD:



Attribute Flow

Disassociation of the Azure AD UPN from the AD DS UPN



Logon ID Change

Active Directory Domain Services on premises:

```
PS C:\Windows\system32> get-aduser jdoe -Properties * |fl userPrincipalName,mail


userPrincipalName : JDoe@corp.pfe.ninja
mail              : JDoe@pfe.ninja
```

Azure Active Directory:

```
PS C:\Users\HybridAdmin\Desktop> Get-MsolUser -UserPrincipalName jdoe@pfe.ninja | fl userPrincipalName,proxyaddresses


UserPrincipalName : JDoe@pfe.ninja
ProxyAddresses    : {smtp:JDoe@pfeninja.onmicrosoft.com, SMTP:JDoe@pfe.ninja}
```

How it Works?

- Takes precedence over but does not eliminate UPN-based authentication
- AD FS queries in-scope forests looking for a single user matching the alternate login ID
- The alternate login ID is not directly used in the logon process, but rather for account lookup
- AD FS Service Account must have read permission to the CN attribute for all users in the lookup forests

Alternate Login ID: Summary

UPN alignment is still the best option

Useful for customers who have no on-premises deployments and no need for hybrid

- For example, customers migrating from Lotus Domino

Verify latest support and clients authentication behavior

Chapter

1.2.5

Application integration in Azure AD

🎯 Describe the authentication and permissions mechanisms used by SaaS applications

Authentication vs. authorization, the same? 🤔

Authentication

AuthN

Authentication is the process of proving that you are who you say you are. The Microsoft identity platform uses the OpenID Connect protocol for handling authentication.

Authorization

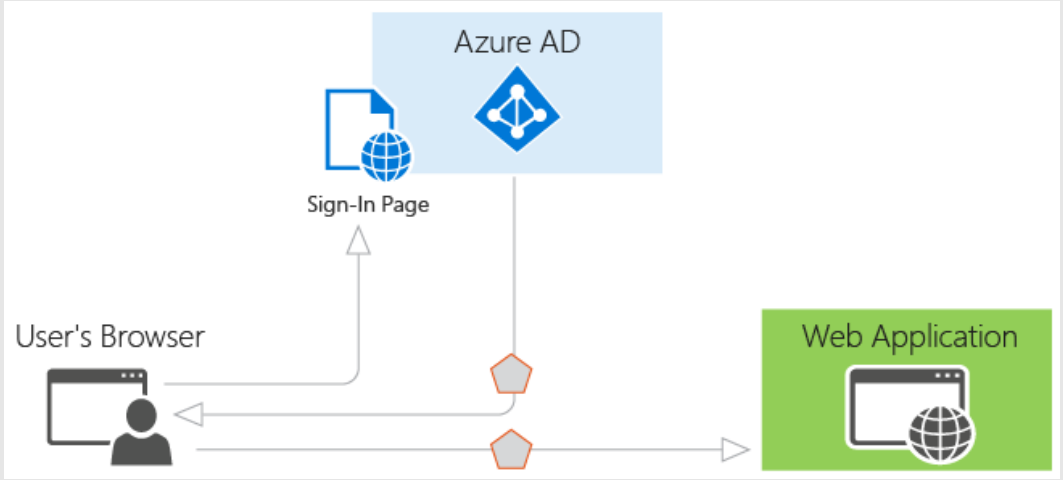
AuthZ

Authorization is the act of granting an authenticated party permission to do something. It specifies what data you're allowed to access and what you can do with that data. The Microsoft identity platform uses the OAuth 2.0 protocol for handling authorization.

Basics of authentication in Azure AD

Azure AD is the identity provider:

- Responsible for verifying the identity of users and applications that exist in an organization's directory
- Issue security tokens upon successful authentication of those users and applications.



An application that wants to outsource authentication to Azure AD must be registered in Azure AD, which registers and uniquely identifies the app in the directory.

Once a user has been authenticated, the application must validate the user's security token to ensure that authentication was successful for the intended parties. Developers can use the provided authentication libraries to handle validation of any token from Azure AD, including JSON Web Tokens (JWT) or SAML 2.0.

What is Modern Auth?

- Combination of authentication and authorization methods between a client and a server
- Security measures that rely on access policies that you may already be familiar with. It includes:
  - Authentication methods: Multifactor authentication (MFA); smart card authentication; client certificate-based authentication
  - Authorization methods: Microsoft's implementation of Open Authorization (OAuth)
  - Conditional access policies: Mobile Application Management (MAM) and Azure Active Directory (Azure AD) Conditional Access

Modern authentication

MSAL

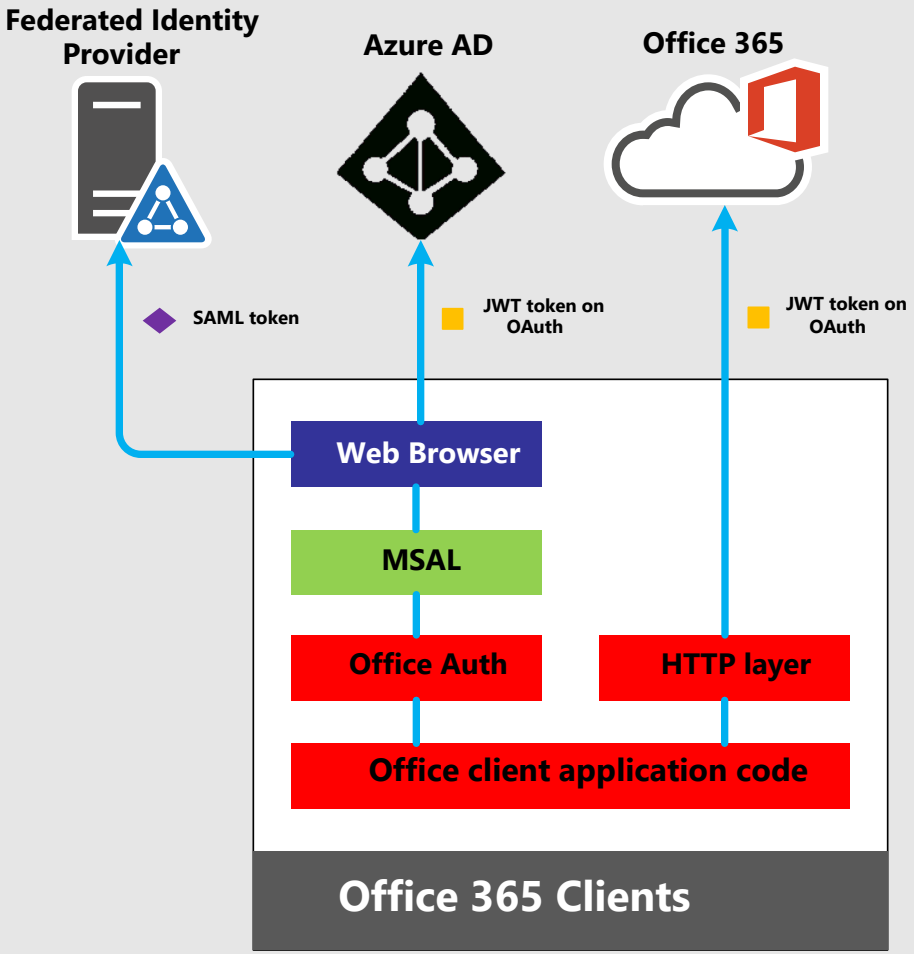
Applications use Microsoft Authentication Library to authenticate from Azure AD

- Use OAuth 2.0
- There is a client making a request for a resource
- There is a resource to which the client needs a specific level of access, and this resource is secured by a directory service
- There is an OAuth connection, in other words, a connection that is dedicated to authorizing a user to access a resource. (OAuth is also known by the more descriptive name, 'Server-to-Server' auth, and is often abbreviated as S2S.)

Enables sign-in features such as:

- Multi-Factor Authentication
- SAML-based third-party Identity Providers with Office client applications
- Removes the need for Outlook to use the basic authentication protocol.

Modern Authentication Flow - basics



Modern AuthN but legacy AuthN?

Legacy means that they support either:

- Microsoft Online Sign-in Assistant (MOS SIA) or basic authentication

The MOS SIA provides end user sign-in capabilities to Microsoft Online Services installing client components that allow common applications to authenticate to Microsoft Online Services

- Can also provide an improved sign-in experience, such that end users can access Microsoft Online Services without having to re-enter their credentials (such as a user name or password).

Basic authentication sends a Base64-encoded string that contains a username and password for the client.

- Base64 is not a form of encryption and should be considered the same as sending the username and password in clear text. If a resource needs to be protected, strongly consider using an authentication scheme other than basic authentication.

Deprecated end of Year 2022

And for What??

FOR SSO of course!! But not only → To apply conditions

MFA, compliance, location, type of device, ….

Users will not be required to sign in again to App when:

User has already signed-in to another Azure AD register App on the same device

App running on a domain-joined Windows device

App running on a Compliance cloud domain-joined Windows 10 device

OAuth vs OpenID vs SAML!

OAuth versus OpenID Connect:

- The platform uses OAuth for authorization and OpenID Connect (OIDC) for authentication.
- OpenID Connect is built on top of OAuth 2.0, so the terminology and flow are similar between the two.
  - You can even both authenticate a user (through OpenID Connect) and get authorization to access a protected resource that the user owns (through OAuth 2.0) in one request

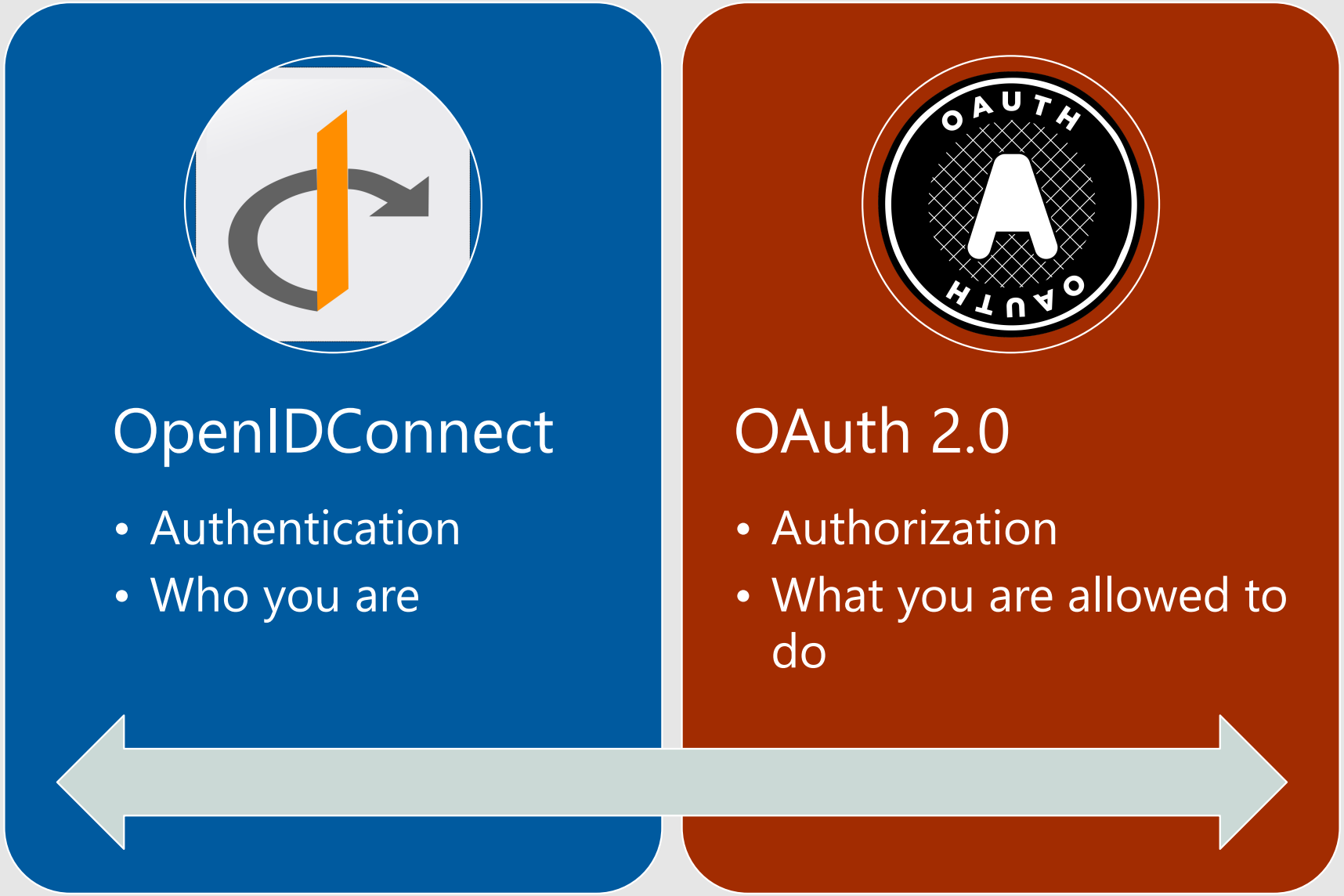
OAuth versus SAML:

- The platform uses OAuth 2.0 for authorization and SAML for authentication.

OpenID Connect versus SAML:

- The platform uses both OpenID Connect and SAML to authenticate a user and enable single sign-on.
  - SAML authentication is commonly used with identity providers such as Active Directory Federation Services (AD FS) federated to Azure AD, so it's often used in enterprise applications.
  - OpenID Connect is commonly used for apps that are purely in the cloud, such as mobile apps, websites, and web APIs.

OpenIDConnect & OAuth 2



OAuth 2.0

OAuth 2.0 Allow third parties to access your resources with limited privileges for a limited period of time without sharing your actual credentials.

Delegated access to subset of data

Limited Scope (Read vs write access)

Standardized method of accessing data

Delegated Authorization

OAuth is not an authentication protocol

OAuth is used for Delegated Authorization

OAuth is like a valet key used with luxury cars 😎:

> *"luxury cars today can come with a valet key. It is a special key you give the parking attendant and unlike your regular key, will not allow the car to drive more than a mile or two. Some valet keys will not open the trunk, while others will block access to your onboard cell phone address book. Regardless of what restrictions the valet key imposes, the idea is very clever. You give someone limited access to your car with a special key, while using your regular key to unlock everything."*

OAuth Client types

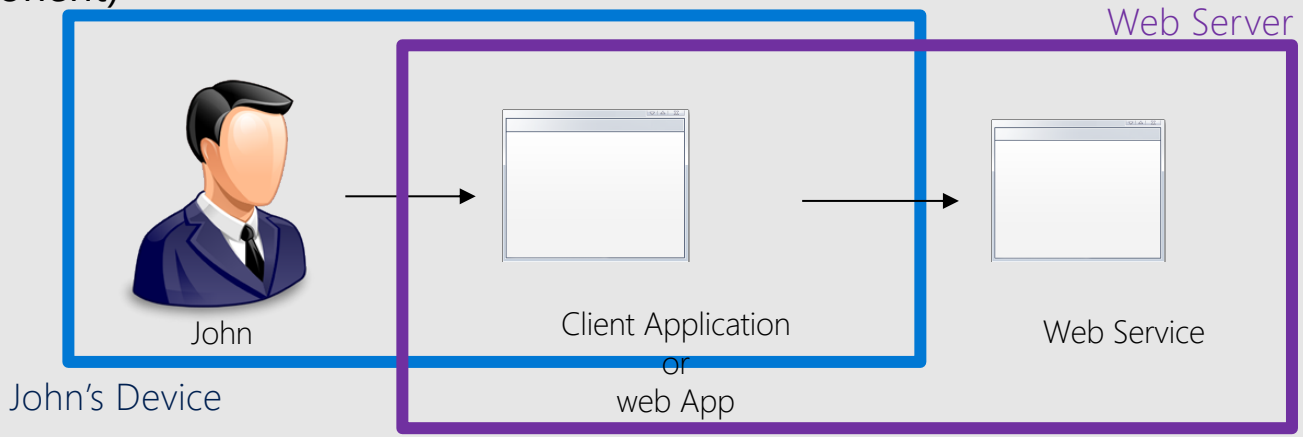
Clients application access a service on behalf of a user, 2 Client app types:

Public client app

- Runs locally on a device. Not trusted to hold a secret.
- Mostly native/desktop applications
  or user-agent-based applications (javascript application running in a browser).

Private/Confidential client app

- Can be trusted with secrets.
- A client capable of keeping a secret (typically server applications, not in end-user's hands)
- Mostly web apps (confidential server-based component)



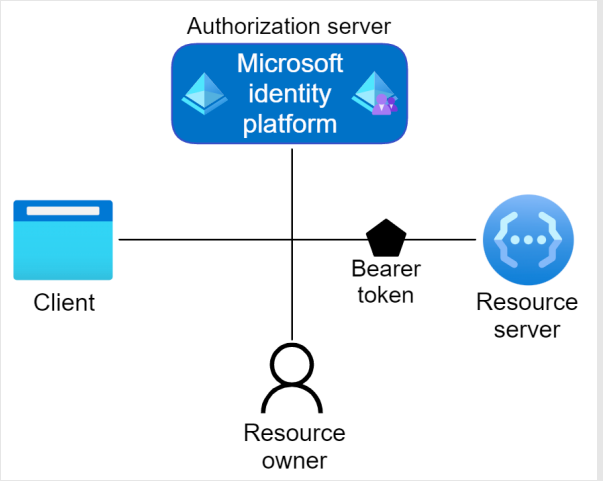
Adv.

Roles in OAuth 2.0

Authorization server

- The Microsoft identity platform itself is the authorization server, called an Identity Provider
- Issue Security Tokens after the AuthN

IdP



Client

- The client is the application requesting access to a protected resource.
- Is NOT the Device

Resource owner

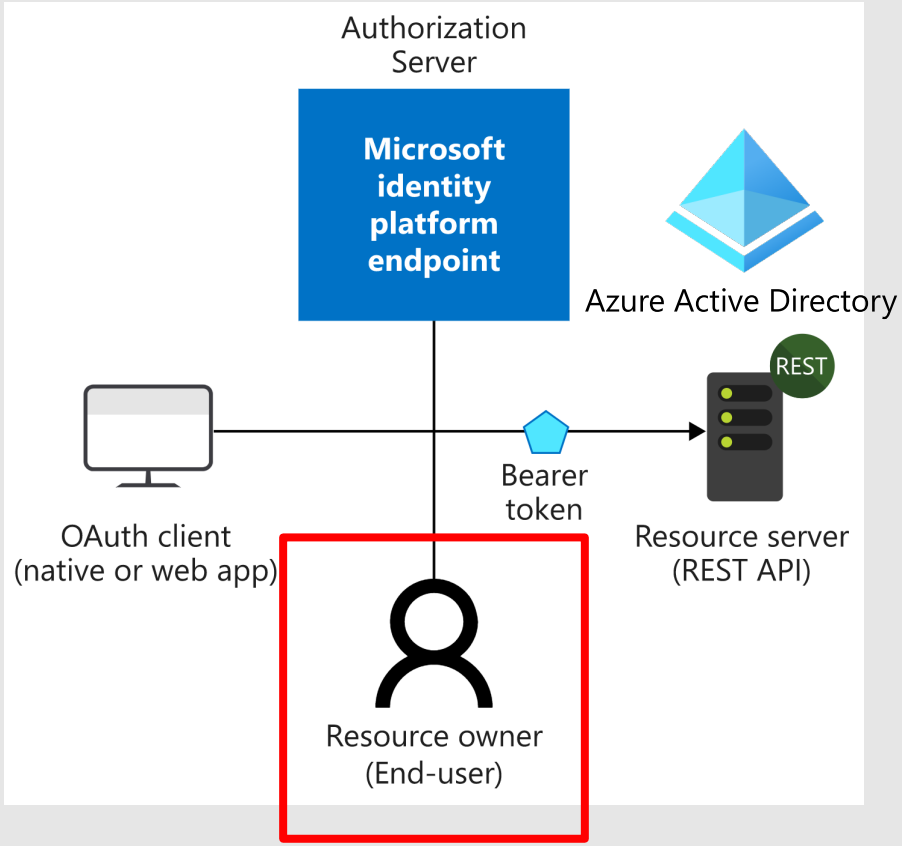
- The resource owner is typically the application user, or the end-user
- The end-user "owns" the protected resource (their data)

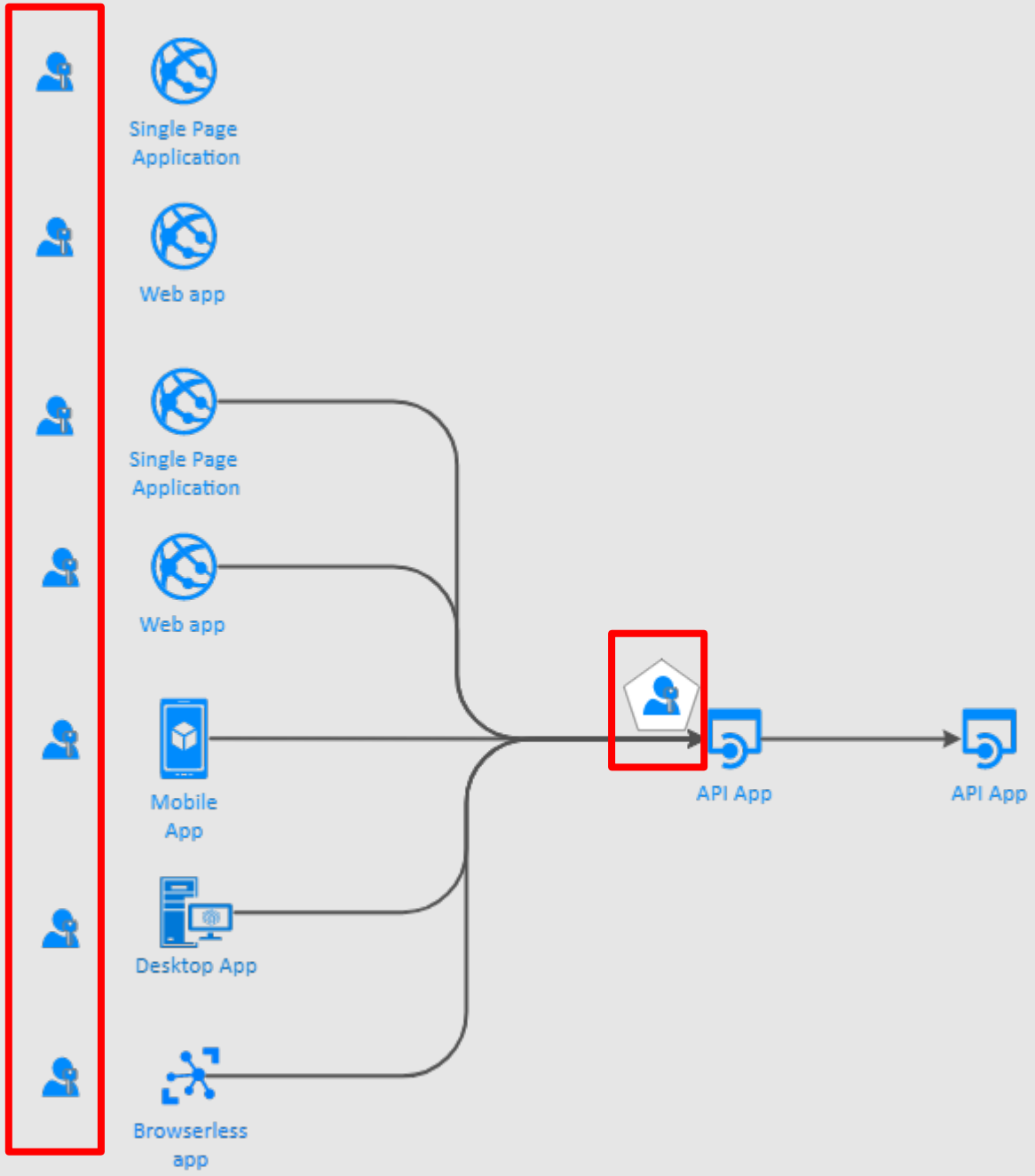
Resource server

- The resource server hosts or provides access to a resource owner's data.

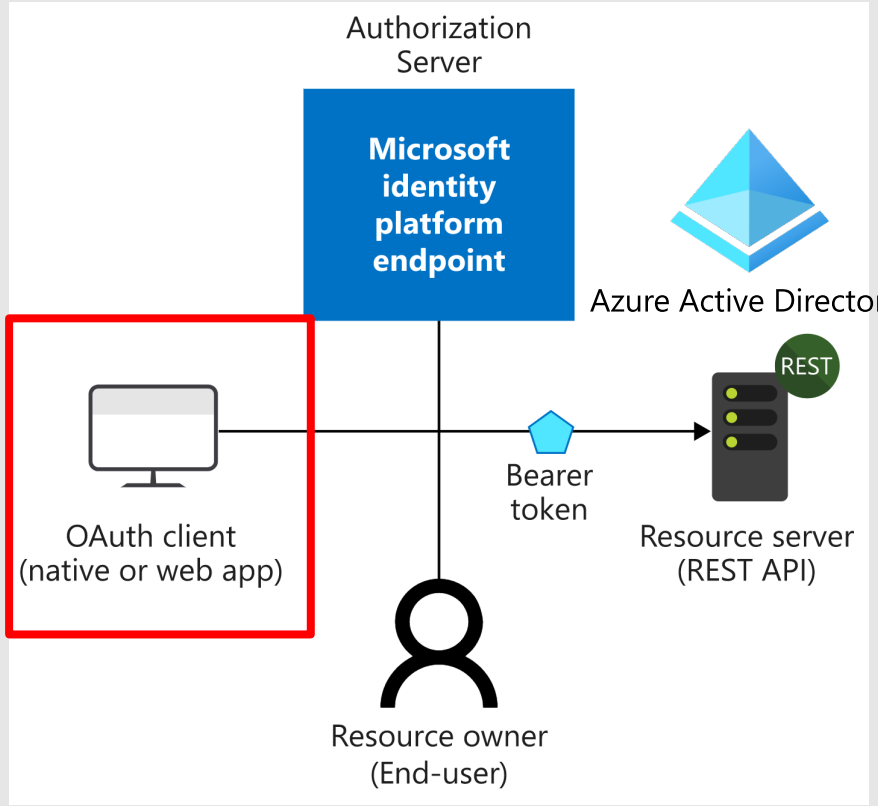
Adv.

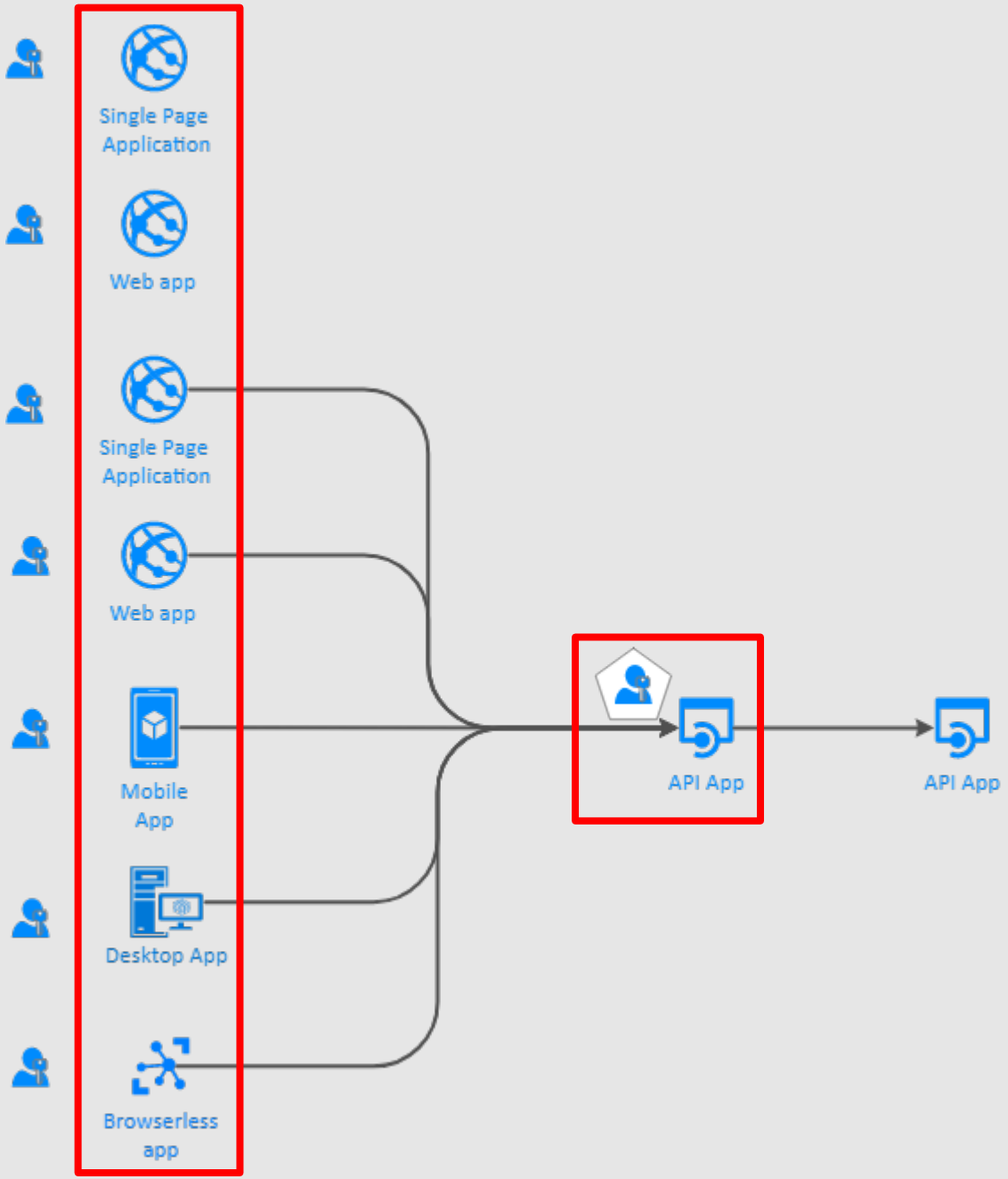
Acquire tokens on behalf of user - Resource Owner





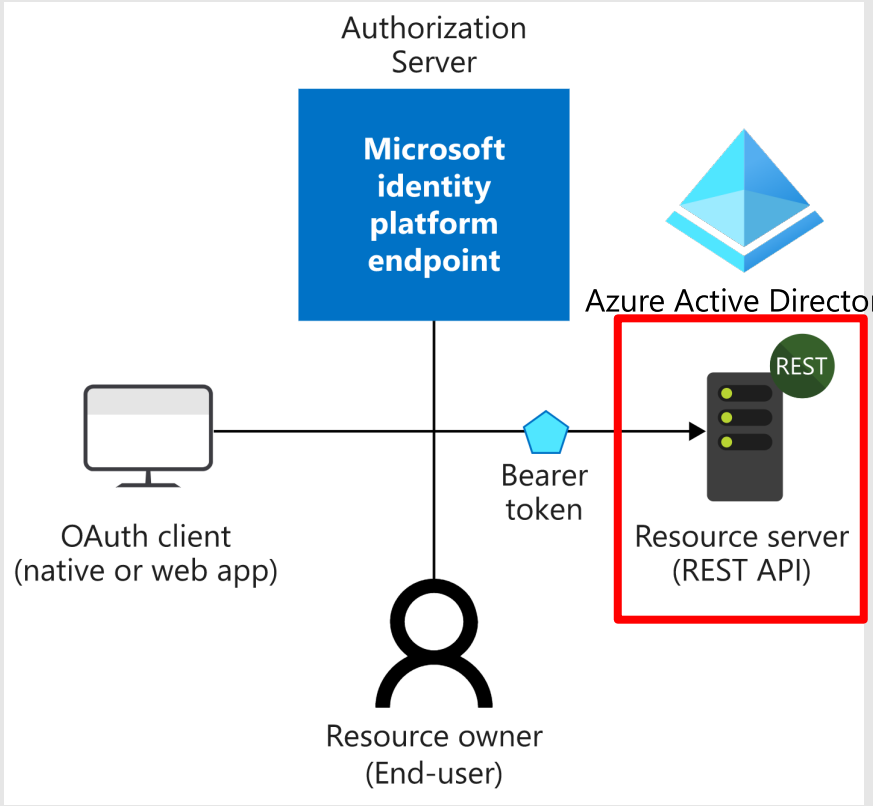
Acquire tokens on behalf of user - OAuth client

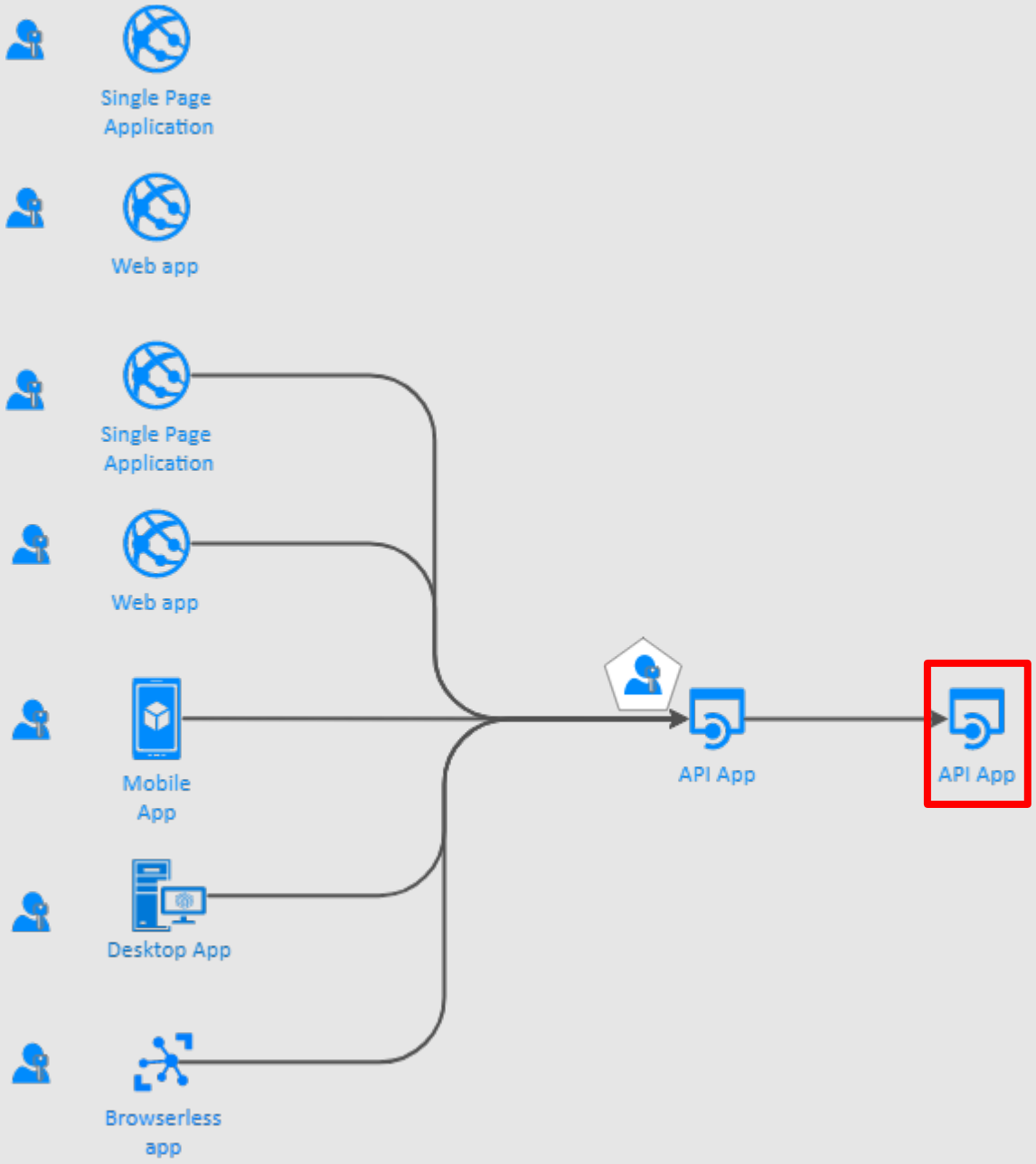




Adv.

Acquire tokens on behalf of user – Resource server





OAuth Client App Profiles

Web Application

- Is a **confidential client** running on a web server
- Resource owners access the client via an HTML user interface rendered in a user-agent on the device used by the resource owner
- The client app credentials as well as any access token issued to the client are stored on the web server and are not exposed to or accessible by the resource owner.

User-agent-based Application

- Is a **public client i**n which the client code is downloaded from a web server and executes within a user-agent (e.g., web browser) on the device used by the resource owner
- Protocol data and credentials are easily accessible (and often visible) to the resource owner
- Since such applications reside within the user-agent, they can make seamless use of the user-agent capabilities when requesting authorization.

Native Application

- Is a **public client** installed and executed on the device used by the resource owner
- Protocol data and credentials are accessible to the resource owner
- It is assumed that any client authentication credentials included in the application can be extracted
- On the other hand, dynamically issued credentials such as access tokens or refresh tokens can receive an acceptable level of protection
- At a minimum, these credentials are protected from hostile servers with which the application may interact
- On some platforms, these credentials might be protected from other applications residing on the same device

Adv.

Client App Registration

For most scenarios, the Client app needs to be registered with the Authorization Server in advance

- Ensure that the client is a known and trusted entity
- Generates a client identifier
- Establish an authentication secret between the Client and Authorization Server

Client must provide

- Client type: Public / Private
- Allowed redirection URI's (where to send the Tokens)
- Other information such as name, website, description, logos, etc.

After registering, the authorization server will provide the client:

- Client ID
- Client Secret

OAuth Endpoints

Two 'Authorization Server' endpoints

Authorization Endpoint

- End-user facing
- User authenticates and grants consent
- This is normally a web page with a login and password box

Token Endpoint

- Used for obtaining an access token
- Typically, a web service

One Client Endpoint

Redirect Endpoint

- Client listens for authorization code
- Used in the authorization code grant scenario

OAuth 2.0 Flows

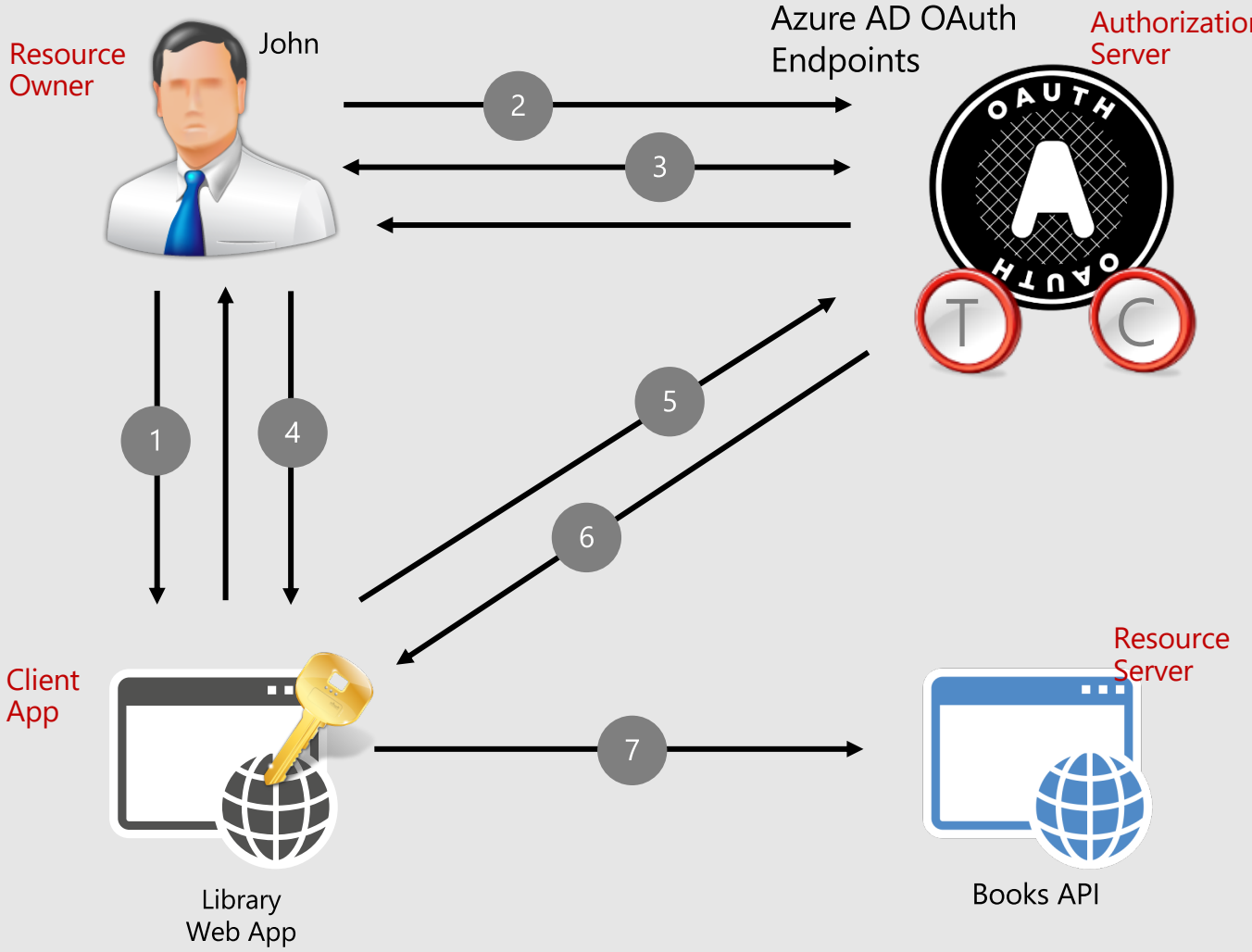
- **Authorization Grant**

  An authorization grant is a "credential" representing the resource owner's authorization (to access its protected resources) used by the client app to obtain an access token.

- **RFC 6749 defines four Authorization Grant Types**

- **Authorization Code**
  - Server-side web applications (e.g. A web application that requires information from O365 API on your behalf)
  - One-time code issued to client / Client redeems code for access token
  - *Client Web app: Request redirects Authorization Request Access Token uses Access Token to access Resource*
- **Implicit**
  - Used for client-side web apps running in the browser (JavaScript Flash Silverlight etc.)
  - Client is untrusted (public)
  - No refresh token issued
- **Resource Owner Password Credentials**
  - Highly-trusted clients such as mobile client applications from the Resource Server (e.g. the authorized Native Mobile client)
  - Client collects username/password from user (resource owner)
  - Then exchange username password for access token (Request token with resource owner credentials)
- **Client Credentials**
  - Clients that can use resources irrespective of Resource Owner authorization – Client to service communication
  - Authenticates the client, not the user!
  - Client receives an access token for itself

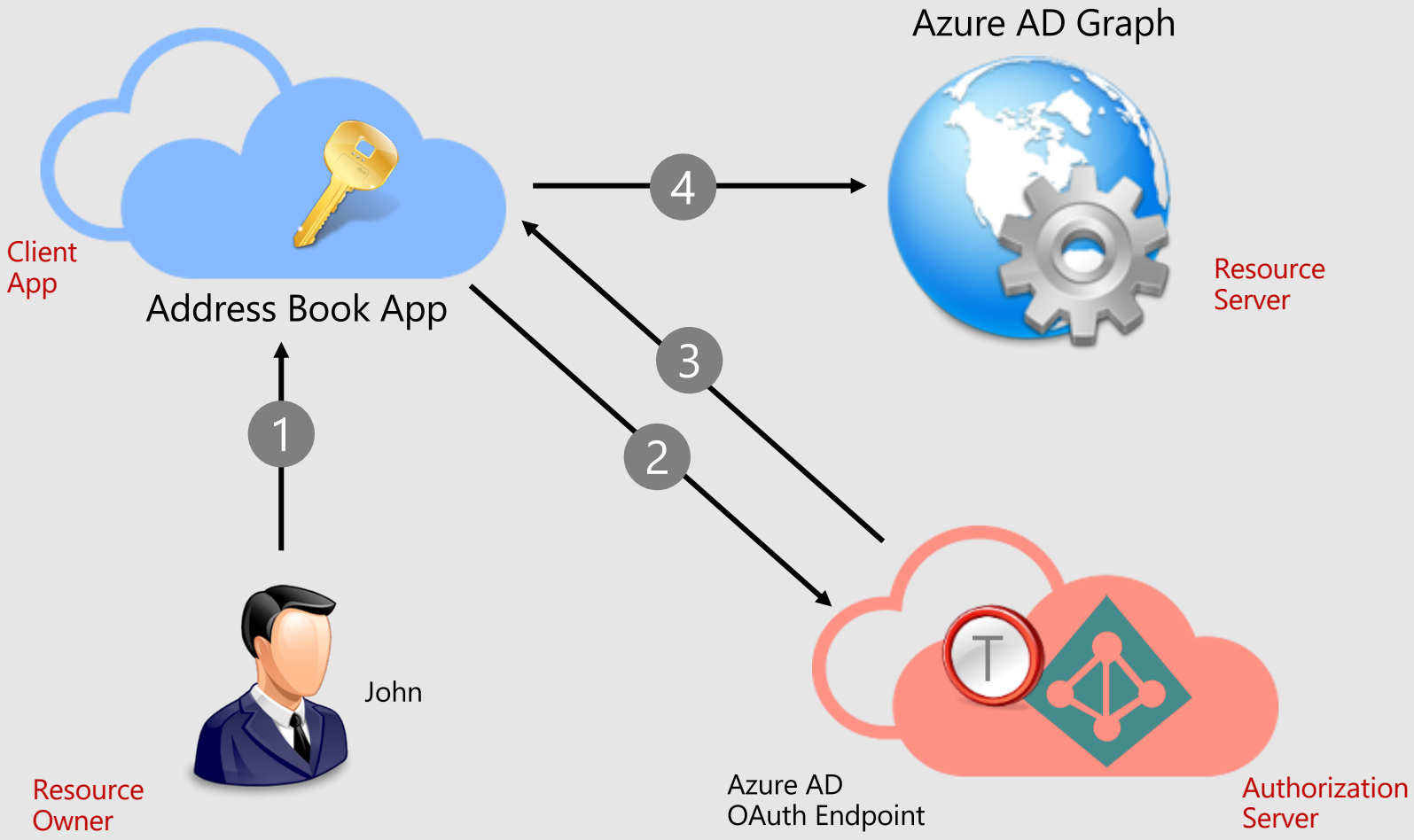
OAuth: Example of Authorization Code Flow



1. John logs into Library client web app and chooses to show his currently checked out books.
2. Library App **redirects** John to OAuth *authorize* endpoint
3. John **authenticates** and grants **consent**
4. OAuth Server redirects John back to Library App with **Authorization Code**
5. Library App requests **A**ccess **T**oken for Books web API from OAuth *token* endpoint
6. OAuth Server returns **Access Token** to Books API
7. Library App uses Access Token to authorize to Books API

Adv.

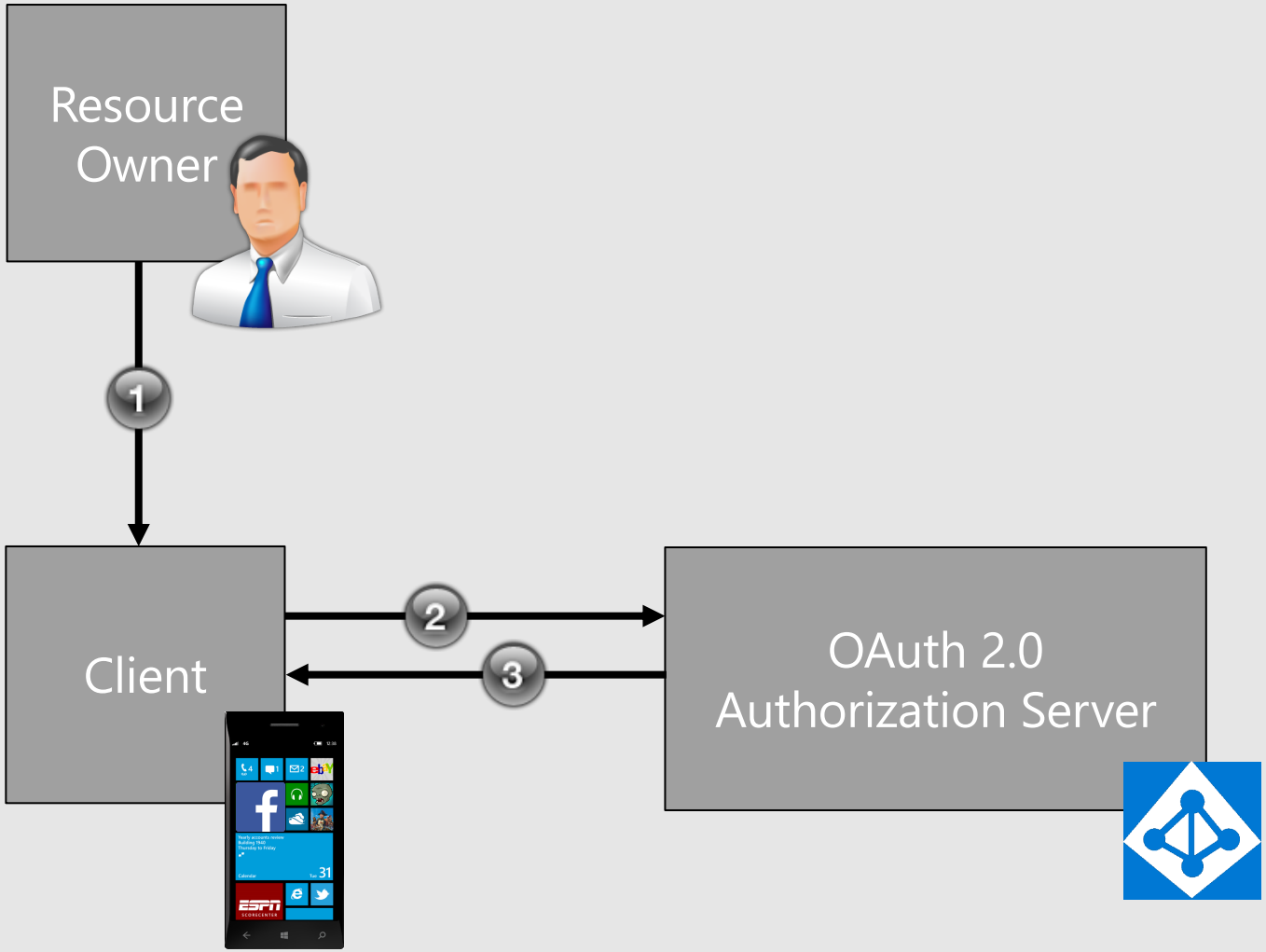
Client Credentials Flow Summary



1. John uses Client app (Address book app)
2. Client App (**using its own creds:** a shared secret or certificate) authenticates to OAuth Server (Azure AD)
3. OAuth Server provides access token to Client
4. Client calls Resource (Web Service) with access token in header

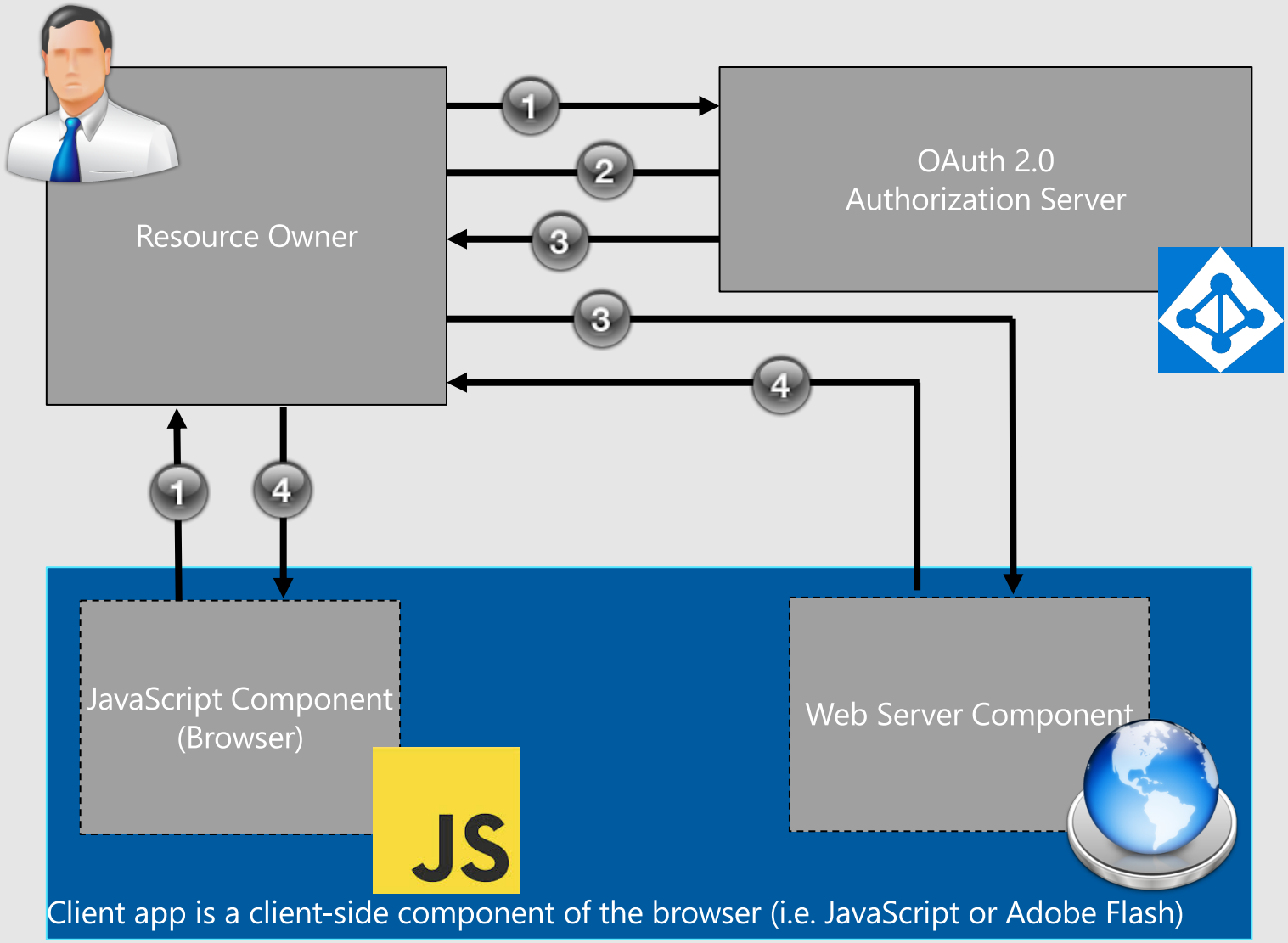
Note: The client credentials grant type MUST only be used by confidential clients.

Resource Owner Password Credentials Flow Summary



1. **Resource Owner gives credentials to Client**
2. Client uses credentials and to request access token from Authorization Server
3. Authorization Server validates the resource owner credentials, and returns access token

Implicit Flow Summary



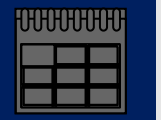
1. Client redirects Resource Owner to Authorization Server.
2. Resource Owner authenticates and grants consent
3. Authorization Server redirects Resource Owner to the Client's Server Component with the **Access Token** in URL fragment
4. Client's Server Component returns script that extracts Access Token from the URL fragment

OAuth2.0 Tokens or Bearer Tokens

Access Token

- Used by a Client to access a protected resource
- Can only be used for a specific combination of user, client and resource
- Contains client, aud, … claims
- Are valid until their expiry
- Cannot be revoked
- If a malicious user has obtained access to a token, he/she can use it

Refresh Token



- Used to get a new access token
- No sign-in action needed as long as token is valid
- Valid for 14 days by default, up to 90 days with continuous use
- Validity is checked on every use
- Can be revoked

OIDC Token

Identity Token

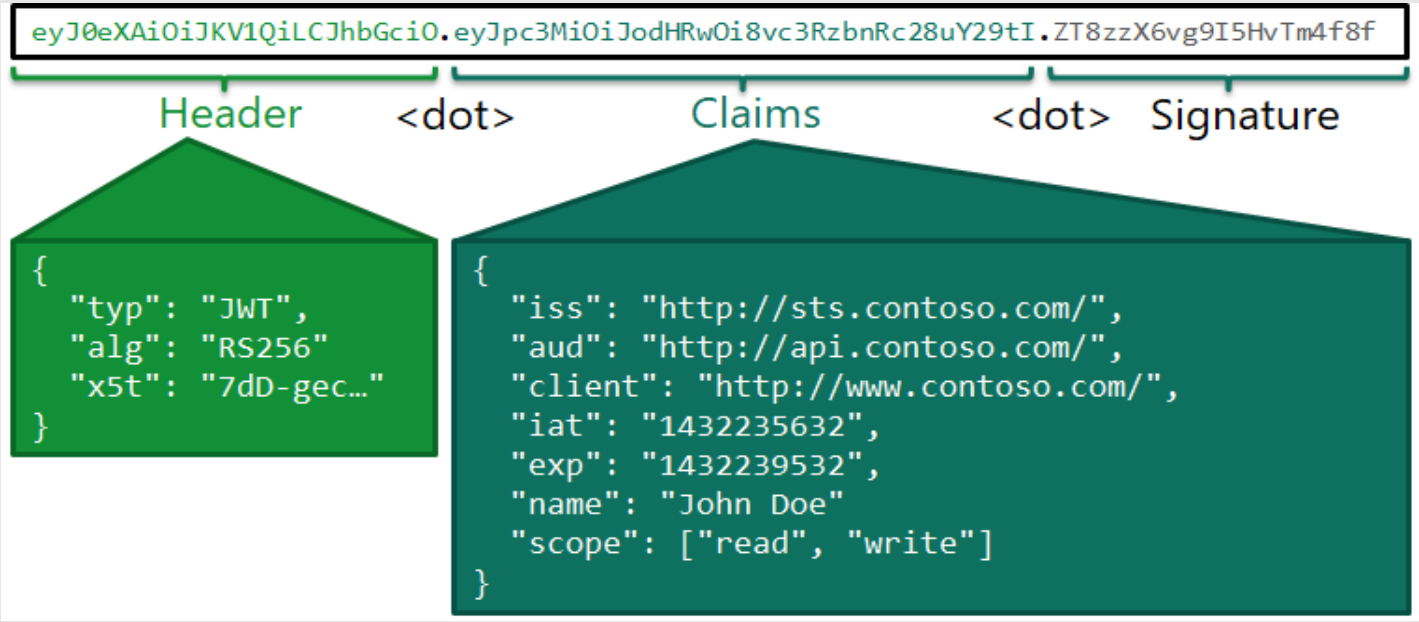
- Asserts the identity of the user (subject)
- Specifies the issuing Authority (iss), Audience (aud), …
- May contain a nonce
- May include additional requested details about the subject (name, email,…)
- Is digitally signed, verified by recipient possible
- May optionally be encrypted for confidentiality

JSON Web Tokens

OAuth protocol uses JWT format for **Access**, **Refresh** and **ID Tokens**

Industry standard for representing claims

Base64 URL encoded JSON with optional signature



OpenID Connect

OAuth 2.0 is purely for authorization, not authentication

- Person granting access might not be the real user (resource owner)
- Does not have a notion of an "identity“
- Access Token contains claims about the delegated access rights

OpenID Connect builds on OAuth 2.0 and adds authentication information

- ID Token: JWT with at least a “sub” claim to identify the end user (“subject”)
- UserInfo Endpoint: get more claims about the end user (JSON/JWT)
- Standardized set of scopes
- Standardized implementation

OpenID Connect Overview

OAuth 2.0 is not intended for authentication

- Recall: OAuth 2.0 - delegate resource access rights to third parties

OpenID Connect defines an identity layer on top of OAuth 2.0

- Uses Two OAuth 2.0 Flows:
  - Authorization Code Flow
  - Implicit Flow
- Adds an ID Token to OAuth 2.0 exchange
- Adds ability to request claims using an OAuth 2.0 access token
- Ratified in Feb. 2014
- Microsoft Co-Authored
- Relies on an additional REST-based UserInfo endpoint (in addition of Authorization and Token endpoints)
  - For getting additional claims about a user
  - Authenticate with Access Token received from OpenID Connect Provider
  - Response returned in JSON.

OpenID Connect

OAuth 2.0 does not tell the client who the user is

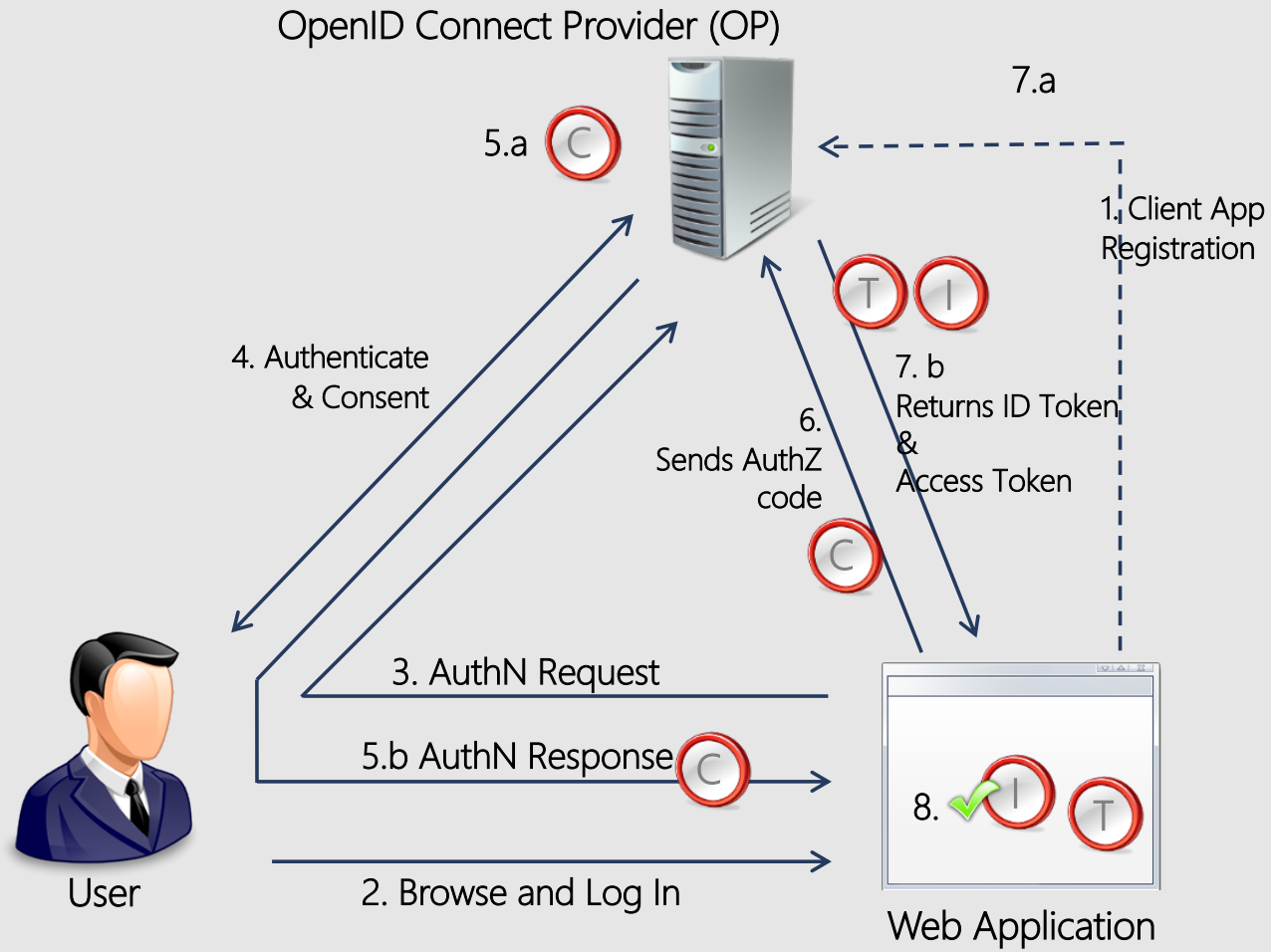
OIDC: add user identity request to OAuth2 request

Becomes pure authentication protocol if access token is **NOT** requested

Replaces WS-Fed as authentication protocol

GET https://login.windows.net/contoso.com/oauth2/authorize?
response_type=id_token%20token&client_id=c5f7b3…

AuthN with the Authorization Code Flow



1. A client registers with the OpenID Connect Provider (OP)
2. User browses to the Web App and initiates Log In
3. Web App redirects User to the OP
4. User authenticates to the OP and gives consent for the Web App to use his identity
5. The OpenID Connect Provider (OP)
   a. builds the Authorization Code [C]
   b. redirects the User back to the Web App with the Authorization Code
6. Web App sends Authorization Code to OP
7. The OpenID Connect Provider (OP)
   a. Creates ID Token [I] and Access Token [T]
   b. And sends back to Web App
8. Web App verifies ID Token

SAML: Security Assertion Markup Language

- Security Assertion Markup Language
- XML protocol
- OASIS standard (V1, v1.1 and V2)
- Flexible and extensible
- Framework to exchange security information between business partners

SAML 2.0 is an XML-based protocol that uses security tokens containing assertions to pass information about a principal (usually an end-user) between an identity provider and a web service (service provider).

SAML 2.0 enables web-based authentication and authorization scenarios including single sign-on (SSO).

ADVANTAGES

- Platform Neutral
- Improves Online Experience for Users
- Supported by many SaaS applications
- Strong commercial and open-source support
- Organizations may extend like Shibboleth 1.3 or Liberty Alliance

SAML: Use Cases and Core Concepts

Assertions:

Authentication, Attribute, and Authorization information

Protocol:

Request and Response elements for packaging assertions

Bindings:

How SAML protocols map onto standard messaging or communication protocols

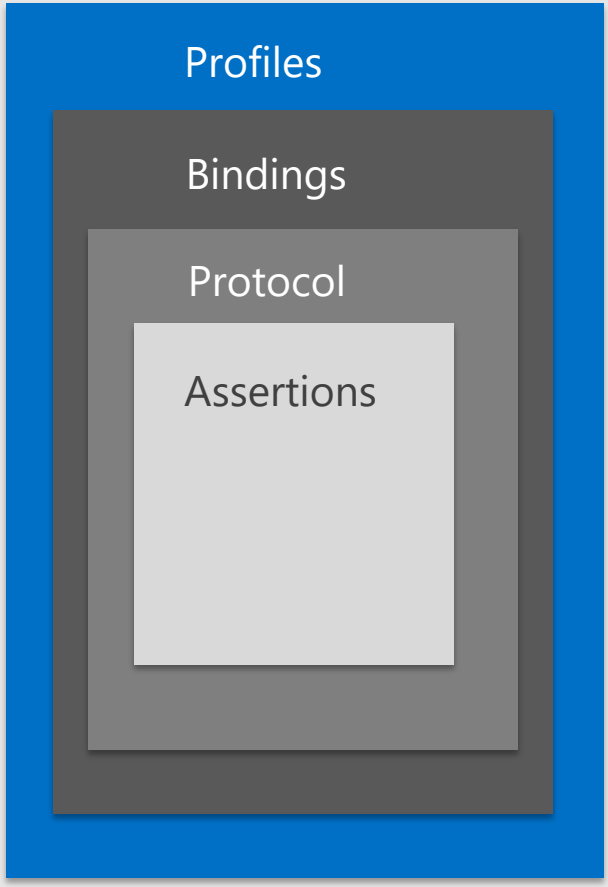
Profiles:

How SAML protocols, bindings, and assertions combine to support a defined use case

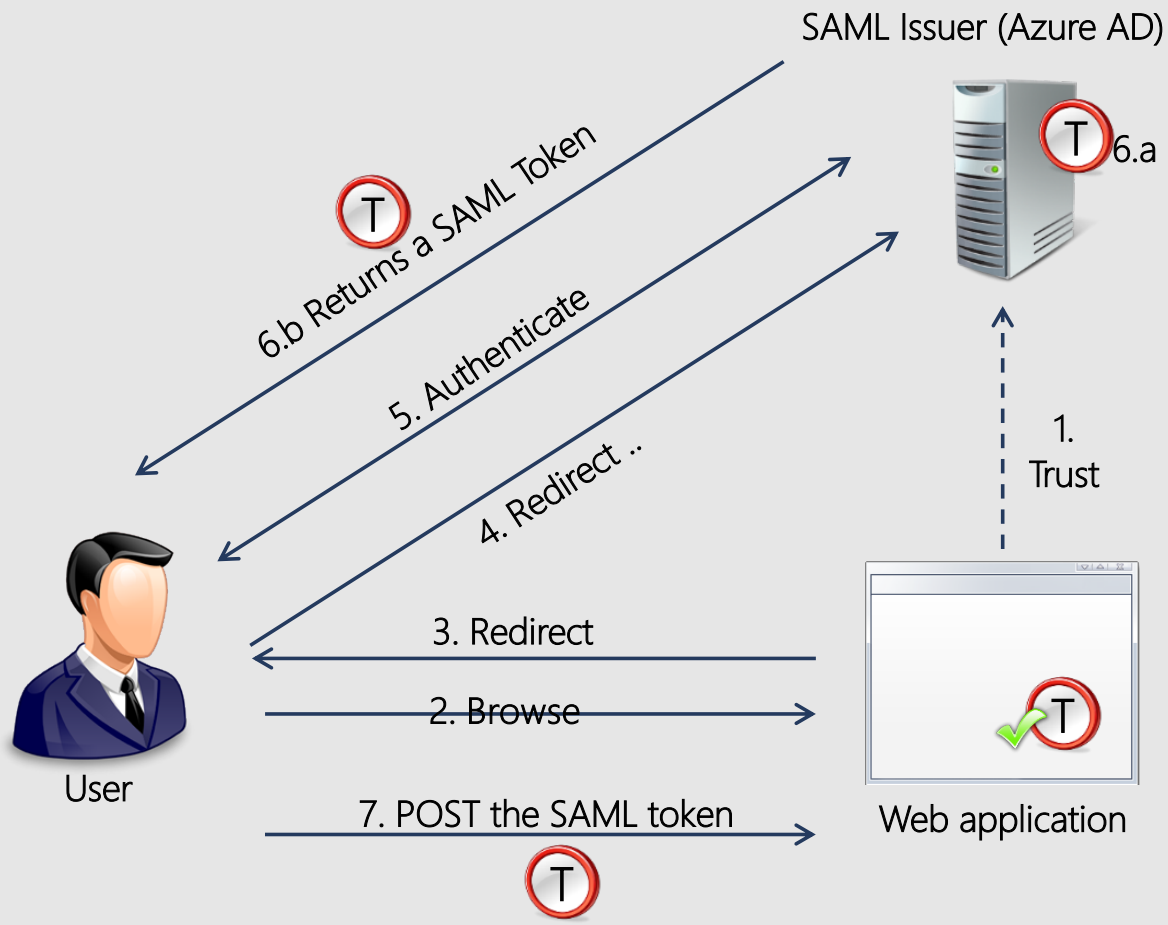
Attribute-based authorization

Web Single Sign-On

Identity Federation



SAML WebSSO profile (scenario) overview



1. A trust is established between the web application and the SAML Issuer
2. User browses to the web application
3. Web application detects that the user is not authenticated and redirects him to the SAML Issuer
4. User automatically browses to the SAML Issuer
5. User authenticates to the SAML Issuer
6. SAML Issuer :
   a. builds the token
   b. and passes it back to the user
7. User POSTs the token to the web application
8. The web application verifies the token

Token Format: Different Flavors of Tokens

SAML – Security Assertion Markup Language

- XML-based
- Very descriptive metadata

JWT – JSON Web Token

- More easily human readable
- Smaller token size

SWT – Simple Web Token

- Form-encoded attribute/value pairs
- Not very common

Kerberos

Some protocols are Token Agnostic

- Some identity protocols are token agnostic
- You can use whatever you want for a token

Differences between Sign-In Protocols

Topic	SAML	OAuth	WS Federation
When to Use	Web Sign-In Usually Third-Party Apps	Rich and Modern Apps Restful Web API	Web Sign-In
Token Formats	XML	JWT	Agnostic (SAML, JWT)
Azure AD Support	Issues SAML 2.0 Tones Support for SAML 2.0	Support for all OAuth 2.0 flows	Yes

Token types

Access Tokens

- Used to access protected resources
- Used for a specific combination of user, client, and resource
- Cannot be revoked and valid until expire

Refresh Tokens

- When resource is accessed, client receives refresh and access token
- Refresh token used to access a new access token when one expires
- Can be revoked

ID Tokens

- Contain profile information about a user. Valid until Expire. Represented as JWT Tokens – contain claims

Session Tokens (SSO)

- Cookie used for Keep-me-Singed-In (kmsi). Can be revoked
- Persistent and on-persistent

Azure AD Application Model

The application model in Azure AD is designed to sustain many different functions:

- It holds all the data required to support authentication at run time.
- It holds all the data for deciding what other resources an application might need to access and whether a given request should be fulfilled and under what circumstances.
- It provides the infrastructure for implementing application provisioning, both within the app developer's tenant and to any other Azure AD tenant.
- It enables end users and administrators to dynamically grant or deny consent for the app to access resources on their behalf.
- It enables administrators to be the ultimate arbiters of what apps are allowed to do and which users can use specific apps, and in general to be stewards of how the directory resources are accessed.

What is Application Consent?

We use permissions and consent every day – think of apps on your cell phone

- The Azure AD Permissions and consent model is very similar

Organizational data is accessed through resource (APIs) that expose **permissions**

Applications need access to **organizational data** at different levels

Those **permissions** can granted through **application consent** by an **admin**, or **end user**

Those **permissions** can be managed over time by an **end user**, **admin**, or **developer**

Terminology

Client

- The **client application** requesting access to data

Resource

- The **application/service** (usually a web API - such GRAPH API) that exposes data

Permission

- The ability for a **client application** to perform some action on some **data** owned by a **resource application**
  - e.g. read a user's OneDrive files through Microsoft Graph

Consent prompt

- The process by which a user is asked to grant an application the **permission(s)** it has requested

Consent grant

- The result of saying "yes" to a consent prompt

Admin(istrative) Consent

- The process by which a company administrator grants an **application** one or more **permissions** that cannot be granted by a regular user. These permissions may:
  - Allow the app to perform high privilege operations- admin-restricted permissions
  - Apply to all users in the organization

User identities vs. application identities

User

User Principal

- Should be used by one user
- Username/password

Service account

- May be used by any given user
- Username/password

User

- Real person
- May or may not have a representation in each identity provider

Application

Service Principal

- May be used by an application hosted anywhere
- Client ID/client secret/certificate

Service account

- May be used by any given user
- Username/password

Web App

- Application
- Can use any identity provider

An App, A service Principal??

To delegate Identity and Access Management functions to Azure AD, an application must be registered with an Azure AD tenant.

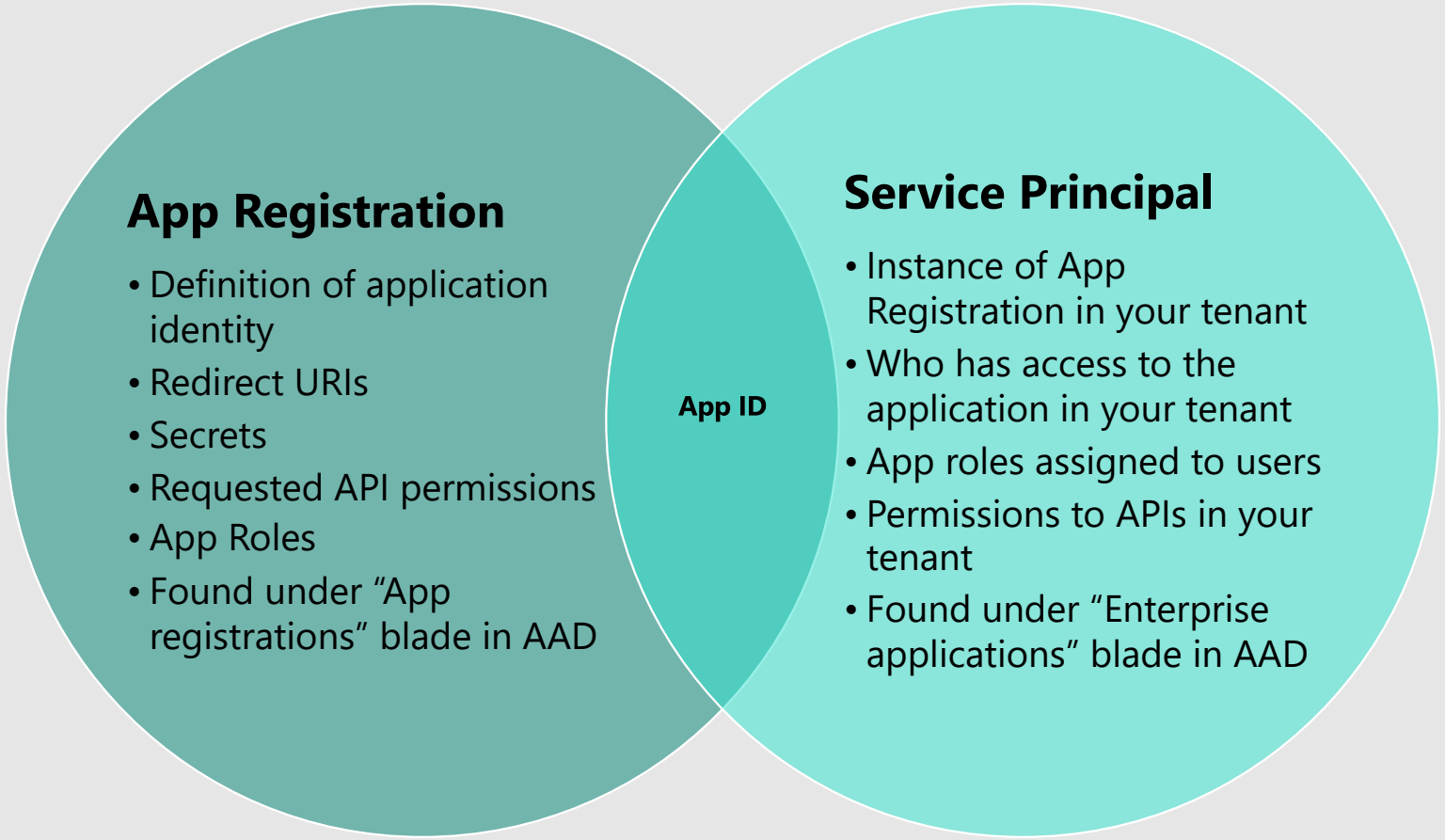
The application object describes three aspects of an application:

- How the service can issue tokens in order to access the application
- Resources that the application might need to access
- Actions that the application can take.

Service **P**rincipal object: 

- Represents an instance of the app in a tenant (the developer's tenant or a customer tenant)
- Used to apply policies
  - Including assigning permissions (ex: allow the app to read your tenant's directory data), the consents
  - Application object & SP are linked. Changes on App are also applied to the associated SP object in your tenant

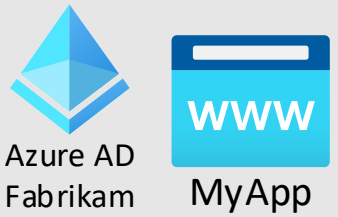
App Registration vs. Service Principal



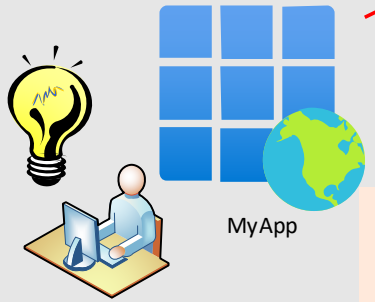
How and why apps are added to Azure AD - Microsoft identity platform | Microsoft Docs

Adv.

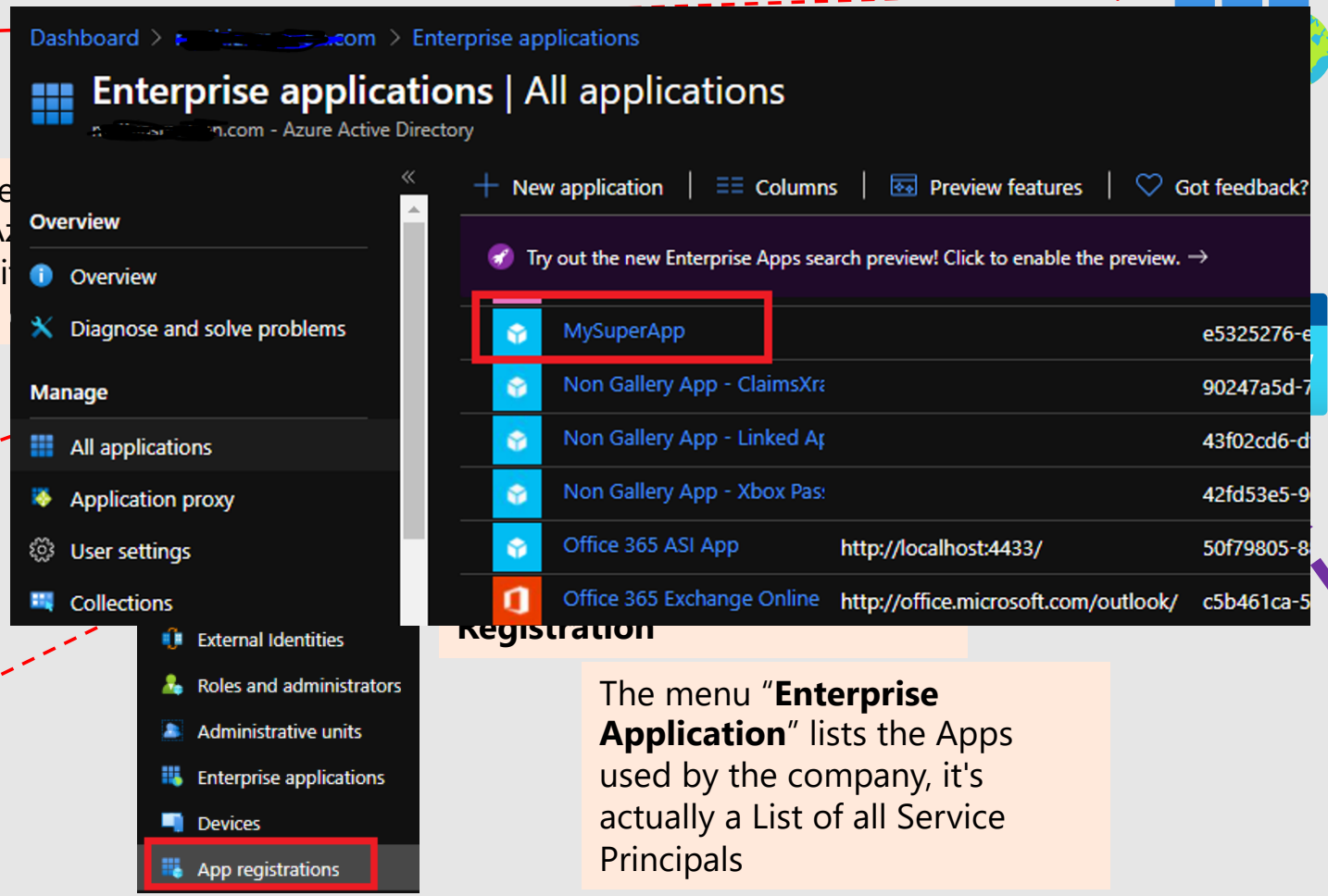
App Registration, Entreprise Apps, Service Principals?



If the App is multite
be used by other A
it is via a "SP" that i
referenced.



An App is developed, we want to make it accessible through Azure AD



The menu "**Enterprise Application**" lists the Apps used by the company, it's actually a List of all Service Principals

An "Application" object is created, it has:

- An AppID
- A Secret
- A URI

A SP object is also automatically created, It references an App for customers and owns;

- Scope of permissions
- Notion of Consent

Adv.

Application model Azure AD objects

3 kind of objects

Application Object	Applications are usually an abstract entity, made of code and resources: **the service principal represents a concrete instance of that abstract entity in a specific directory.**
Service Principal Object	In traditional Active Directory, every entity that can be authenticated is represented by a *principal*. That's true for users, and that's true for applications—in the latter case, we speak of *service principals*. **At deployment time a given Application object can be used as a blueprint to create a ServicePrincipal representing a concrete instance of an application in a directory. It's that ServicePrincipal that is used to define what the app can actually do in that specific target directory, who can use it, what resources it has access to, and so on**
OAuth2PermissionGrants Object	Azure AD maintains another collection of entities, named ***oauth2PermissionGrants***, which records which clients (the applications) have access to which resources (other SaaS services e.g. AAD) and with what permissions (e.g. read user data). Critically, oauth2PermissionGrants also records which users that consent is valid for. **The oauth2PermissionGrants object represents the OAuth 2.0 delegated permission scopes that have been granted to an application (represented by a service principal) as part of the user or admin consent process.**

Service principals


```
{
  "odata.metadata": "https://graph.windows.net/identitydemoxyz.onmicrosoft.com/$me
  "odata.type": "Microsoft.DirectoryServices.ServicePrincipal",
  "objectType": "ServicePrincipal",
  "objectId": "f87517c6-fb9c-4088-9e3e-51b373080aad",
  "deletionTimestamp": null,
  "accountEnabled": true,
  "appDisplayName": "Contoso's Awesome Web App!",
  "appId": "60786a5a-f86a-4fd9-954c-7f6ee14b568a",
  "appOwnerTenantId": "faaac0a7-4362-435a-80e2-a4d32d2953a6",
  "appRoleAssignmentRequired": false,
  "appRoles": [],
  "displayName": "Contoso's Awesome Web App!",
  "errorUrl": null,
  "homepage": "https://awesome.contoso.com",
  "keyCredentials": [],
  "logoutUrl": null,
  "oauth2Permissions": [
    {
      "adminConsentDescription": "Allow the application to access Contoso's Awesom
      "adminConsentDisplayName": "Access Contoso's Awesome Web App!",
      "id": "be42c386-5749-4a6e-9134-b94386bf9d94",
      "isEnabled": true,
      "type": "User",
      "userConsentDescription": "Allow the application to access Contoso's Awesome
```


Registered applications get a ServicePrincipal object in the tenant's directory

OAuth2PermissionGrants are assigned to ServicePrincipal to dictate what they can access


```
"clientId": "f87517c6-fb9c-4088-9e3e-51b373080aad",
"consentType": "AllPrincipals",
"expiryTime": "2016-06-13T08:00:50.7550519",
"objectId": "xhd1-Jz7iECePlGzcwgKrRwgXJDjr-VBgkIle5FGYv0",
"principalId": null,
"resourceId": "905c201c-afe3-41e5-8242-257b914662fd",
"scope": "User.Read",
"startTime": "0001-01-01T00:00:00"
```

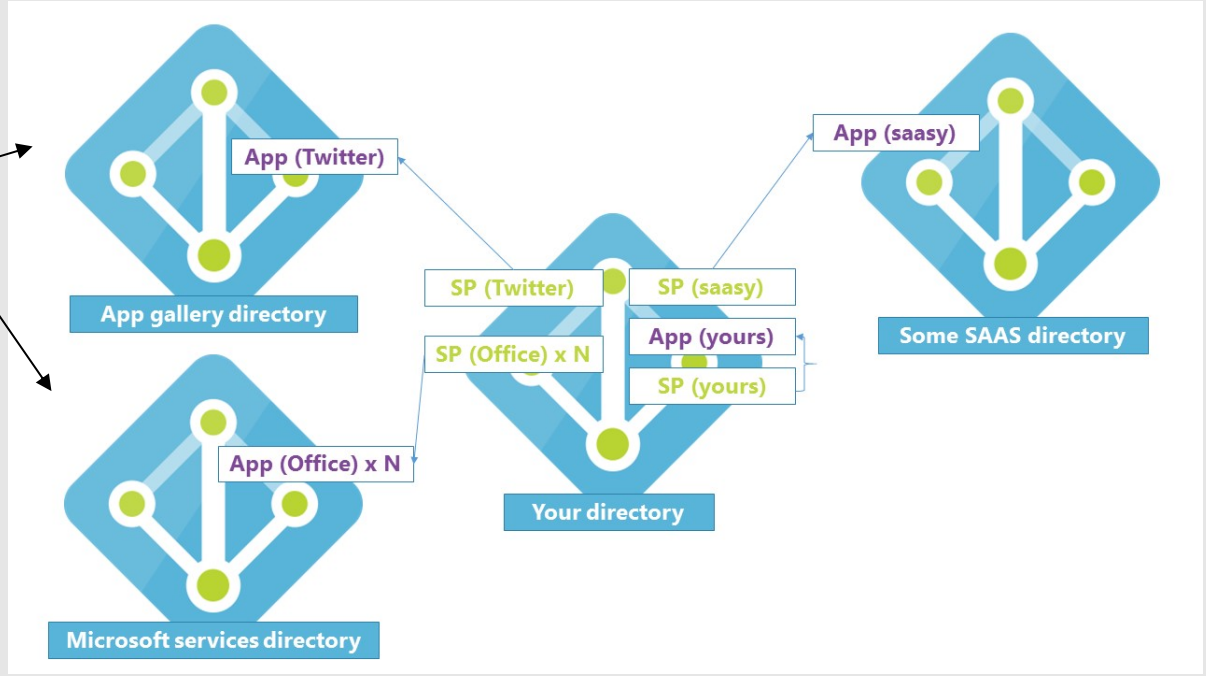
Microsoft apps / Gallery apps /Saas apps directories

Microsoft maintains two directories internally (on the left) it uses to publish applications.

- One for Microsoft Apps (Microsoft services directory)
- One for pre-integrated 3rd Party Apps (App Gallery directory)

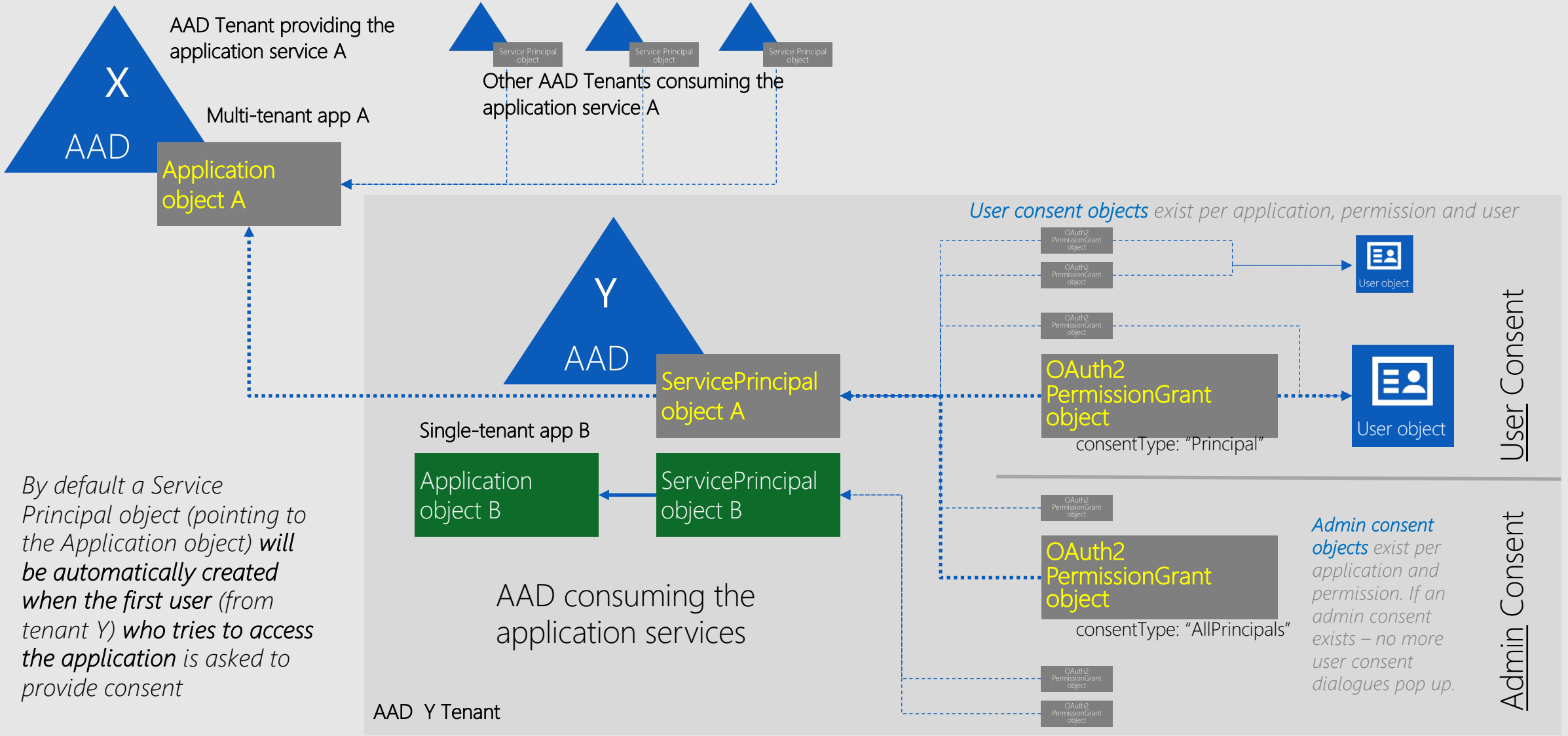
Application publishers/vendors who integrate with Azure AD are required to have a publishing directory. (Some SAAS Directory).

- Applications that you add yourself include:
- Apps you developed (integrated with AAD)
- Apps you connected for single-sign-on
- Apps you published using the Azure AD application proxy.



Adv.

AAD Objects Relationships



By default a Service Principal object (pointing to the Application object) ***will be automatically created when the first user*** *(from tenant Y)* ***who tries to access the application*** *is asked to provide consent*

I want more PRT informations

It’s a key artifact of Azure AD authentication on Windows 10 or newer, Windows Server 2016 and later versions, iOS, and Android devices.

It is a JSON Web Token JWT

It is used during Azure AD app AuthN

14d

Contains Claim:

- Device ID: A PRT is issued to a user on a specific device.
  - deviceID determines the device the PRT was issued to the user on.
  - used to determine authorization for Conditional Access based on device state or compliance.
- Session key: The session key is an encrypted symmetric key, generated by the Azure AD authentication service, issued as part of the PRT.
  - The session key acts as the proof of possession when a PRT is used to obtain tokens for other applications.

Continuously renewed as long as the user actively uses the device

PRT takeaways

Every credential used to unlock has its own PRT

PRT holds AuthN methods for strongest AuthN performed by native app (browsers user WAM in read-only mode)

- Sometimes wrongly referred to as “adding MFA claims”

When PRT is involved you will see fewer prompts than you’d expect

- PRT refreshes and device unlock are considered a prompt – proof of presence in NIST

No prompt = worsened security is a misconception/FUD

Refreshes every device unlock but not more often than 4h

Microsoft Graph API, what's it?

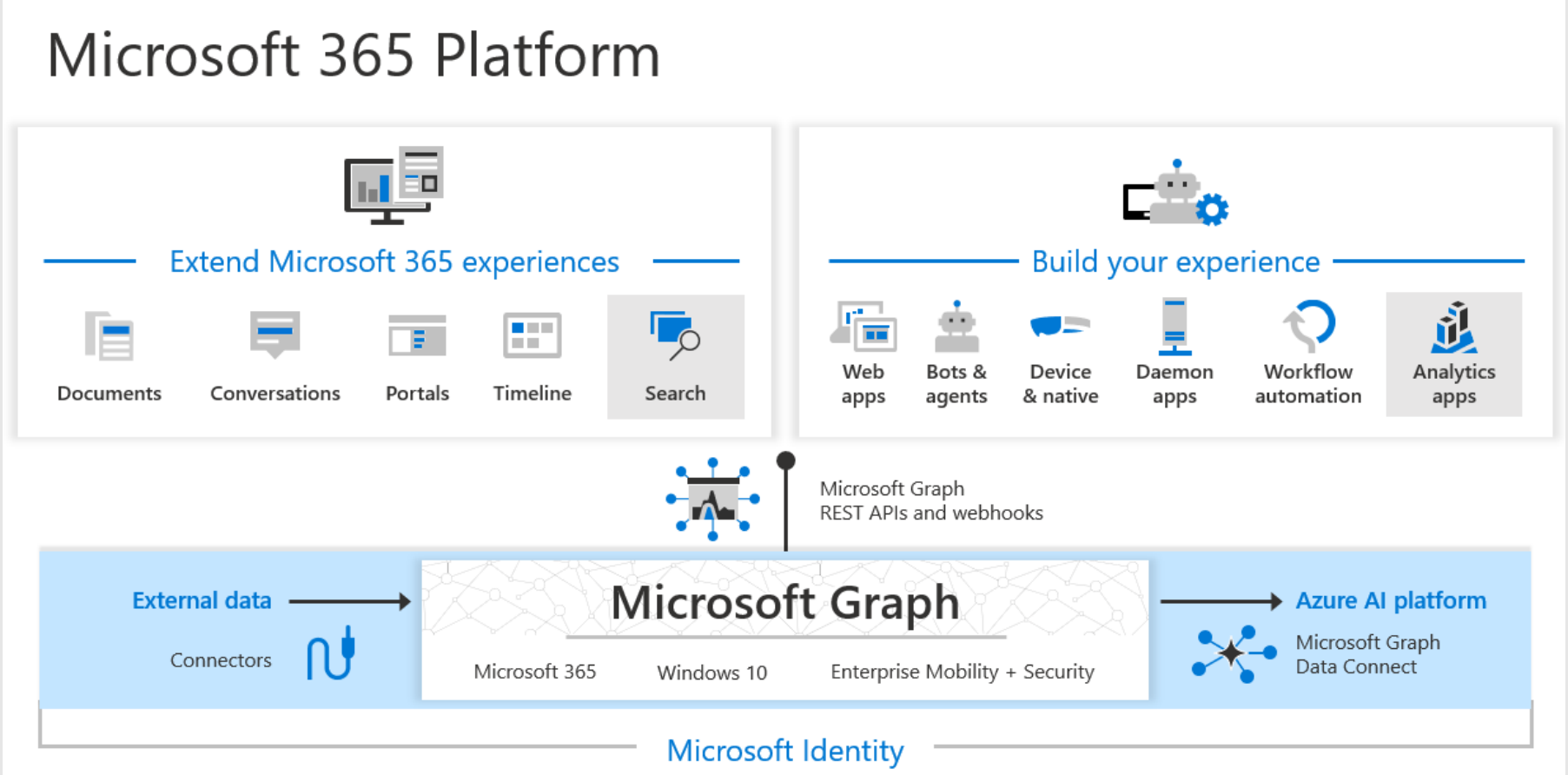
The Microsoft Graph API is a RESTful web API that enables you to access Microsoft Cloud service resources

With MS Graph API, you can:

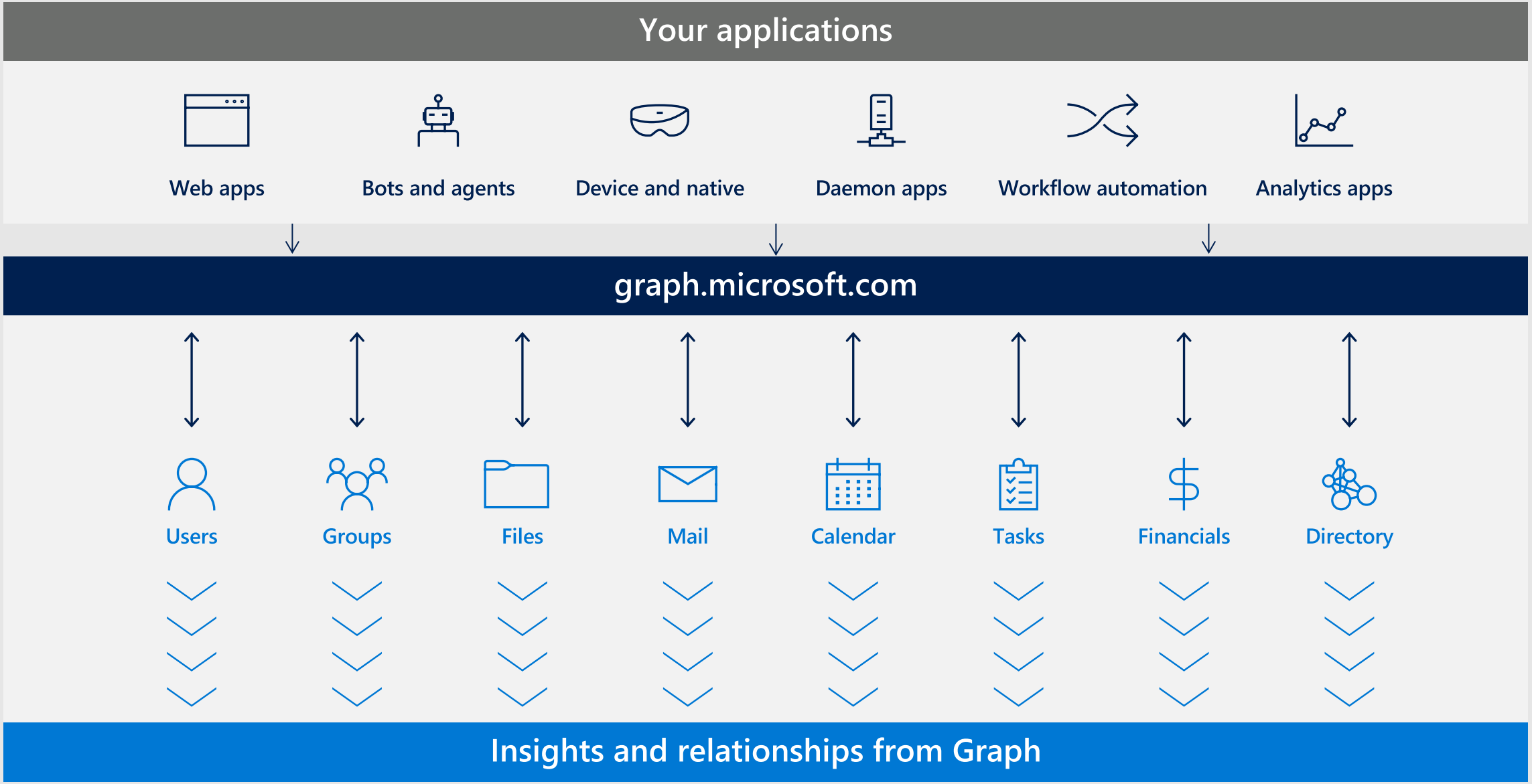
- Get
- List
- Create
- Modify
- Delete
- Update

By using several Language:

- C#, Python, Powershell, …

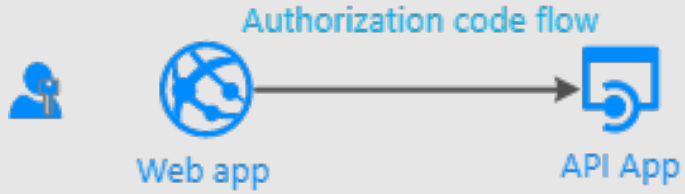


Microsoft Graph: gateway to your data in the Microsoft cloud



Adv.

Access Microsoft Graph API



User

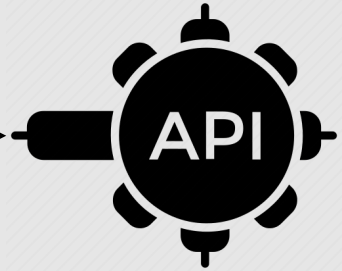
0. Request access token

Service Principal
Postman

Service Principal
Graph API

POSTMAN

2. API validates token &
authorizes action



Graph API

1. Submit request to REST API,
passing access token

AAD Identity management via Azure AD Graph API

Azure Active Directory Graph API provides programmatic access to Azure Active Directory through Odata REST API endpoints

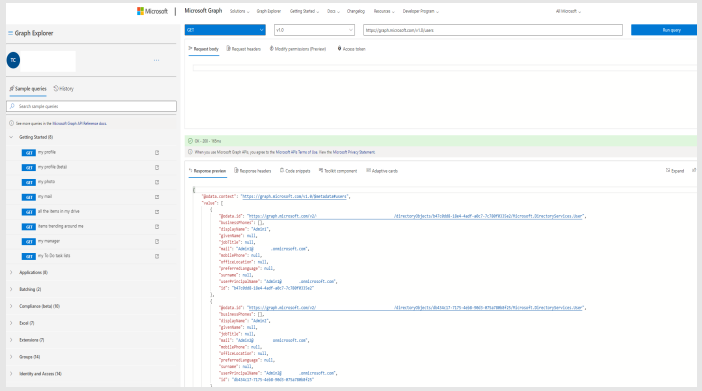
- just as ADSI or ADO.NET provides for Active Directory.

Developers/Applications can execute create, read, update, and delete (CRUD) operations on Azure Active Directory objects such as users and groups.

Graph Explorer to be replaced in the future by Microsoft Graph

Enables the following:

- Line of Business Applications (LOB)
- Multi-tenant apps that require Azure AD Access
- Creating Reusable features that require Azure AD Access



Chapter

1.2.6

Administration of Azure AD

🎯 Describe the administrative roles in Azure AD

Azure AD Roles, not Groups??

In Azure AD, Roles are not provided by groups (not only 😉)

Azure AD roles allow you to grant granular permissions to your Account, abiding by the principle of least privilege

- Work with **P**rivileged **I**dentity **M**anagement (grant just-in-time access to your administrators)

PIM

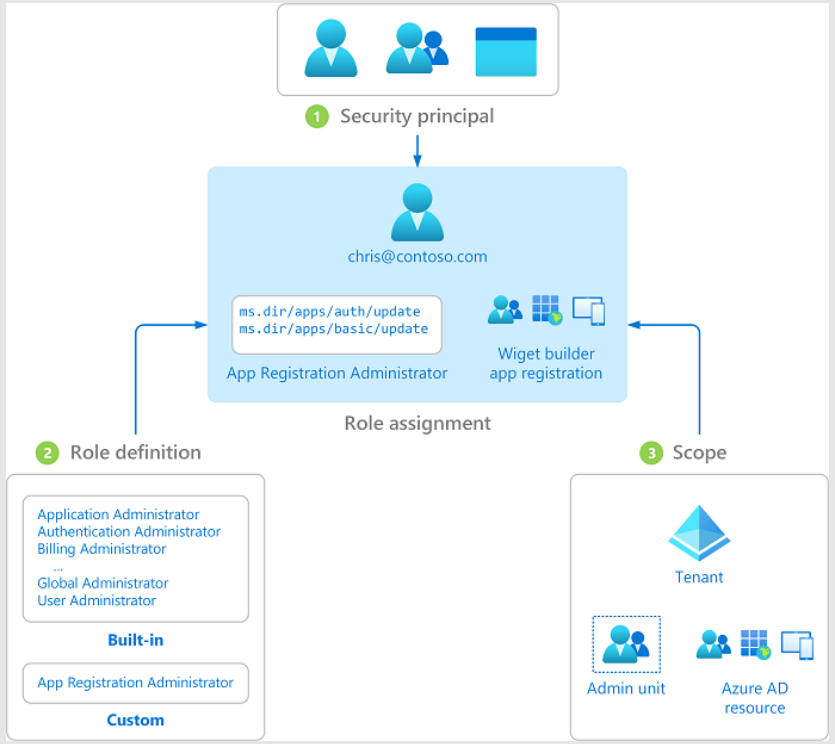
More than 70 administrator roles available, like:

- Global Administrator
- Conditional Access Administrator
- Device Administrator
- Directory Readers / Writers
- And many others

Role-Based Access Control (RBAC) – Roles in Azure AD

Azure AD supports 2 types of roles definitions:

- A collection of permissions
- Built-in roles
  - created by Microsoft that can't be changed
- Custom roles
  - Created and managed by your organization.



At different scopes:

- Organization-wide
- At an Administrative unit
  - Container of resources that can be used for delegating administrative permissions over subsets of users and applying policies to a subset of users
- At an Azure AD Resource

Azure AD Roles (not exhaustive)

Tenant and Security Admin.	Users and Group Admin.	Devices Admin.	Data and Compliance	Microsoft 365 Service Workloads Admin.	Dynamics Admin.	Readers and Support Admin.	Applications Admin.	Other Admin.
Billing Admin	Helpdesk Admin	Intune Admin	Compliance Admin	Exchange Admin	Dynamics 365 Admin/CRM Admin	Global Reader	Application Admin	B2C User Flow Admin
License Admin	Password Admin	Cloud Device Admin	Compliance Data Admin	SharePoint/One Drive Admin		Security Reader	Cloud Application Admin	B2C User Flow Attribute Admin
Global Admin	User Admin	Device Admin	Device Admin	Skype Admin		Reports Reader	Office Apps Admin	B2C IEF Keyset Admin
Privileged Role Admin	Privileged Auth Admin	Desktop Analytics Admin	Customer Lockbox Access Approver	Teams Admin		Directory Reader	Application Developer	B2C IEF Policy Admin
Security Admin	Auth Admin	Printer Admin	Azure Information Protection Admin	Teams Communication Admin		Message Center Privacy Reader	Azure DevOps Admin	Ext Identity Provider Admin
Conditional Access Admin	Group Admin	Printer Technician		Teams Communication Support Specialist		Message Center Reader		*Directory Writers*
	Guest Inviter			Power BI Admin		Service Support Admin		*Directory Synchronization Accounts*
				Kaizala Admin				
				Power Platform Admin				
				Search Admin				
				Search Editor				

Azure Identity Management

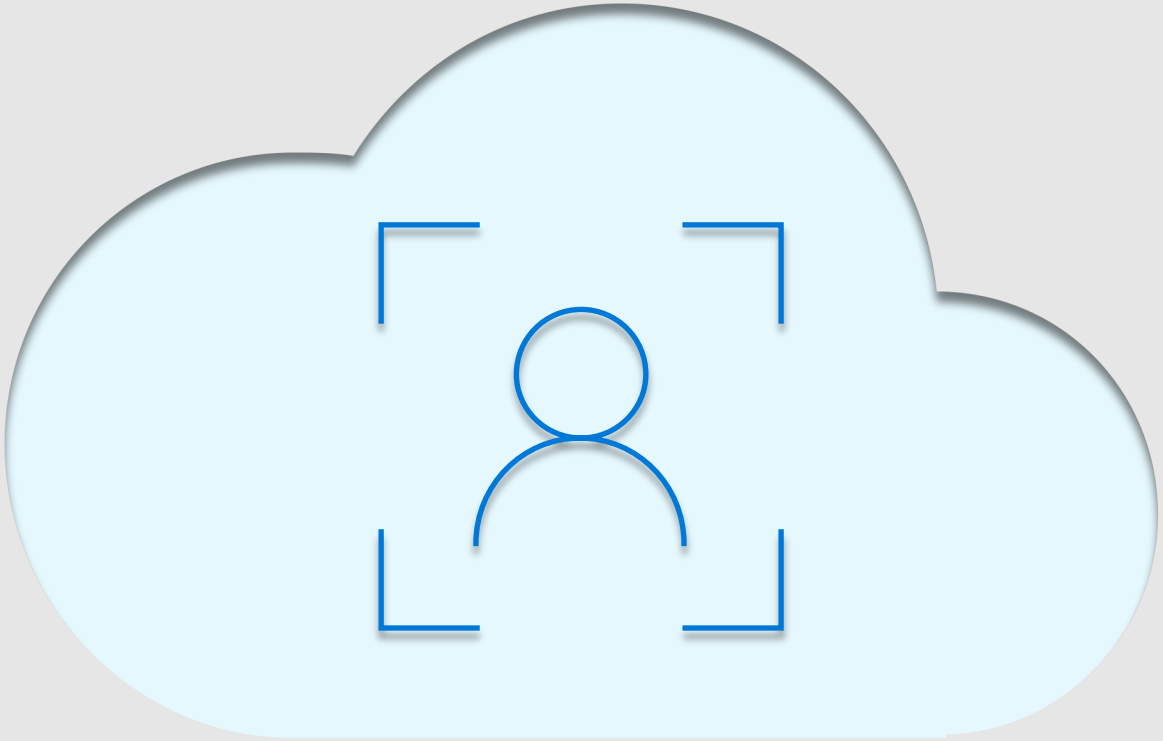
Here are some common tasks:

Users	Groups	Other management tasks
Add new users	Manage groups	Assign licenses
Manage profiles	Manage group members	Manage Reports (sign-ins, Audit, users at risk, etc.)
Share Accounts	Manage group owners	Manage devices
Assign users to administrative roles	Manage group membership	Manage applications
Add guest users (B2B)	…	…
Manage passwords		
…		

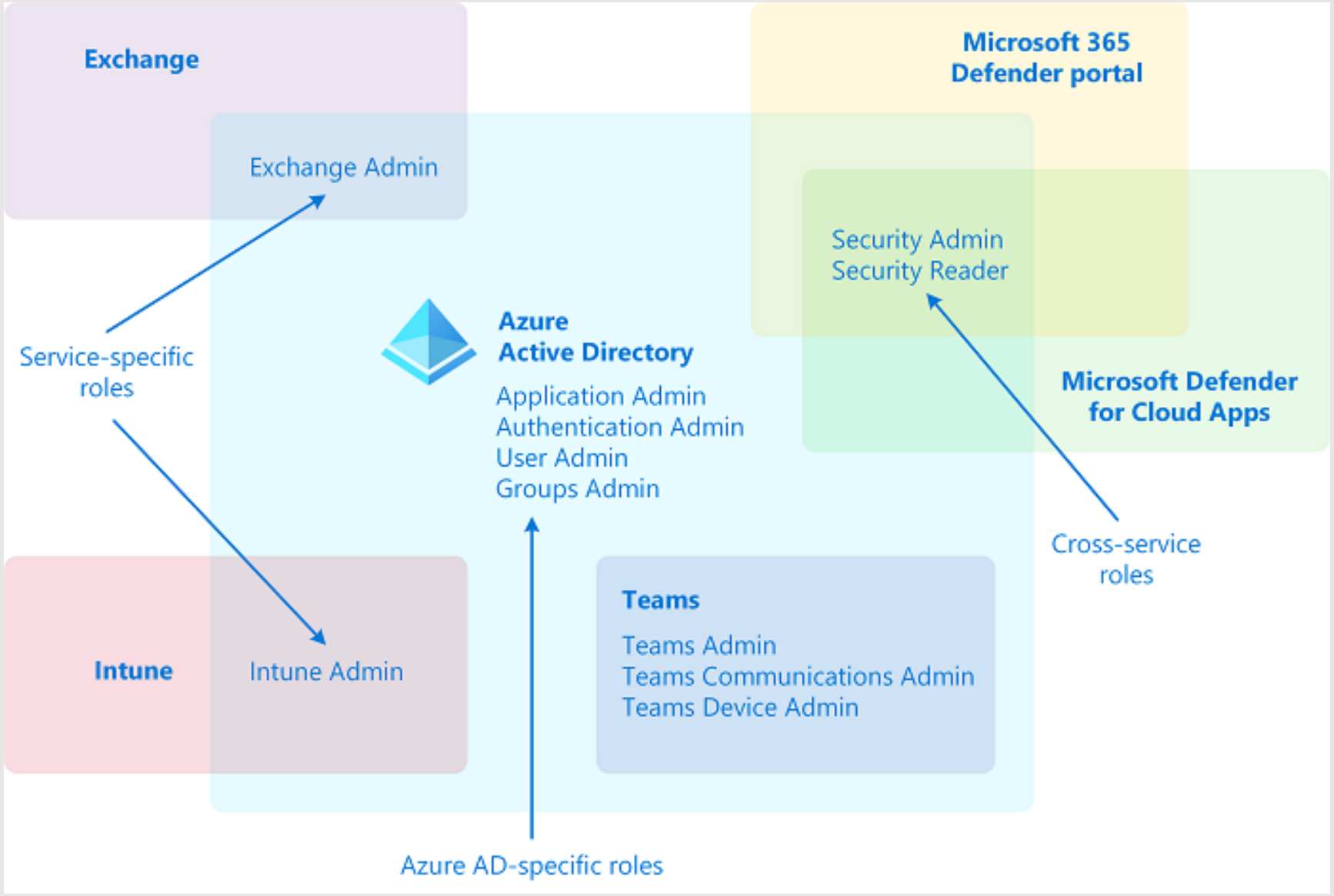
Azure AD Custom Roles

With custom roles you can:

- Limit access even more than built-in RBAC roles;
- Grant granular access;
- Customize access for specific users;
- Can be created through PowerShell, Microsoft Graph, and Azure Portal;
- Can be assigned through Azure Portal;



Azure AD Roles categories



What Are Administrative Units (AUs)?

An administrative unit is an Azure AD resource that can be a container for other Azure AD resources (users, groups & Devices).

Administrative units allow you to grant admin permissions that are restricted to a department, region, or other segment of your organization that you define.

Administrative Unit Management

- Creation/Deletion of Administrative Units
- Assignment of Users and Groups too
- Assignment of scoped role membership

How it’s works?

1. A user (or SP) acquires a token to the Microsoft Graph endpoint.
2. The user makes an API call to Azure AD via Microsoft Graph using the issued token.
3. Depending on the circumstance, Azure AD takes one of the following actions:
    1. Evaluates the user’s role memberships based on the wids claim in the user’s access token.
    2. Retrieves all the role assignments that apply for the user, either directly or via group membership, to the resource on which the action is being taken.
4. Azure AD determines if the action in the API call is included in the roles the user has for this resource.
5. If the user doesn't have a role with the action at the requested scope, access is not granted. Otherwise, access is granted

Roles in the Access Token

JWTs (JSON Web Tokens) are split into three pieces:

- Header - Provides information about how to validate the token including information about the type of token and how it was signed.
- **Payload - Contains all of the important data about the user or app that is attempting to call your service.**
- Signature - Is the raw material used to validate the token.

Wids claim are in the Payload part

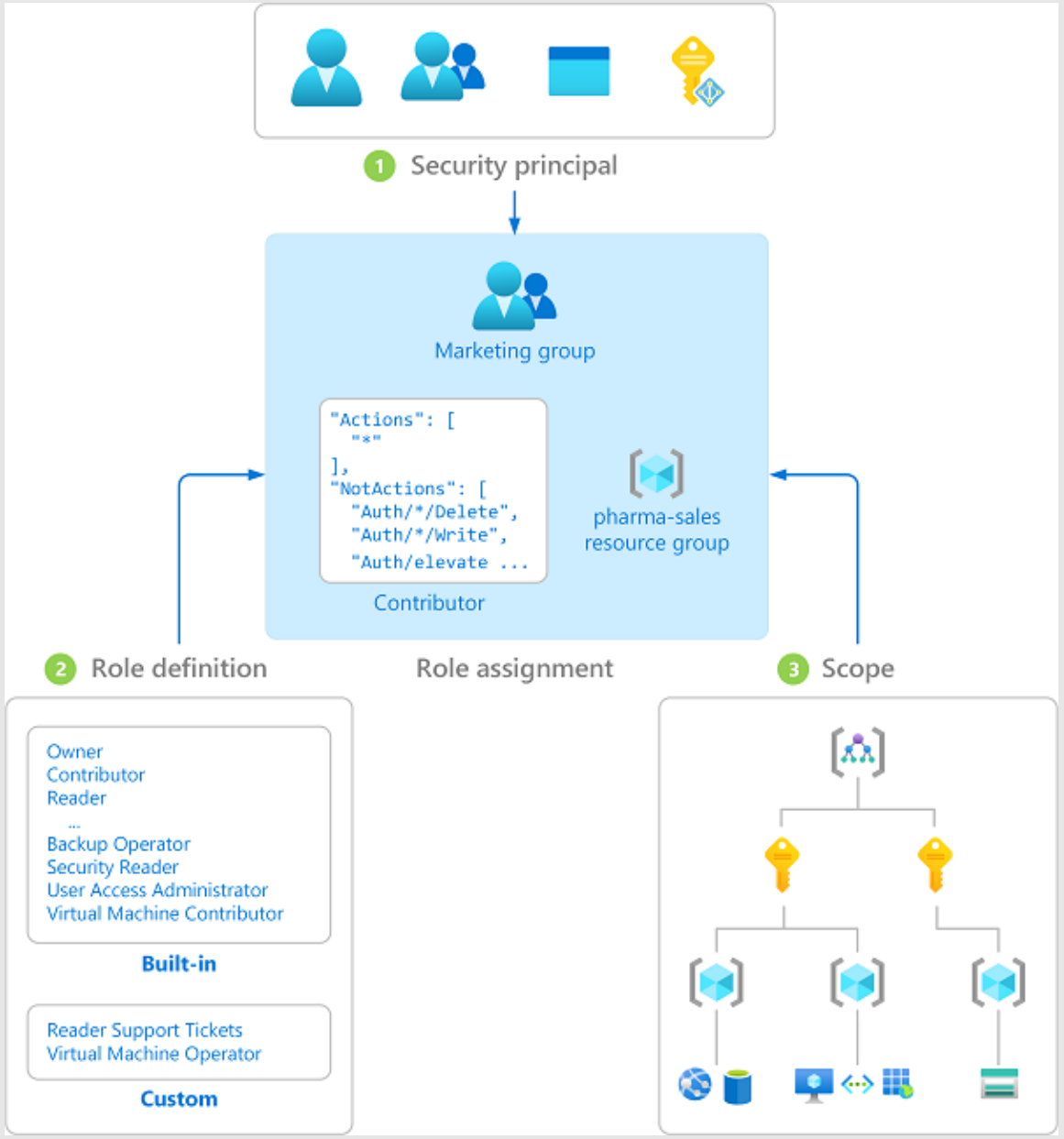
wids	Array of RoleTemplateID GUIDs	Denotes the tenant-wide roles assigned to this user, from the section of roles present in Azure AD built-in roles. This claim is configured on a per-application basis, through the groupMembershipClaims property of the application manifest. Setting it to "All" or "DirectoryRole" is required. May not be present in tokens obtained through the implicit flow due to token length concerns.

Roles? Only in Azure AD?

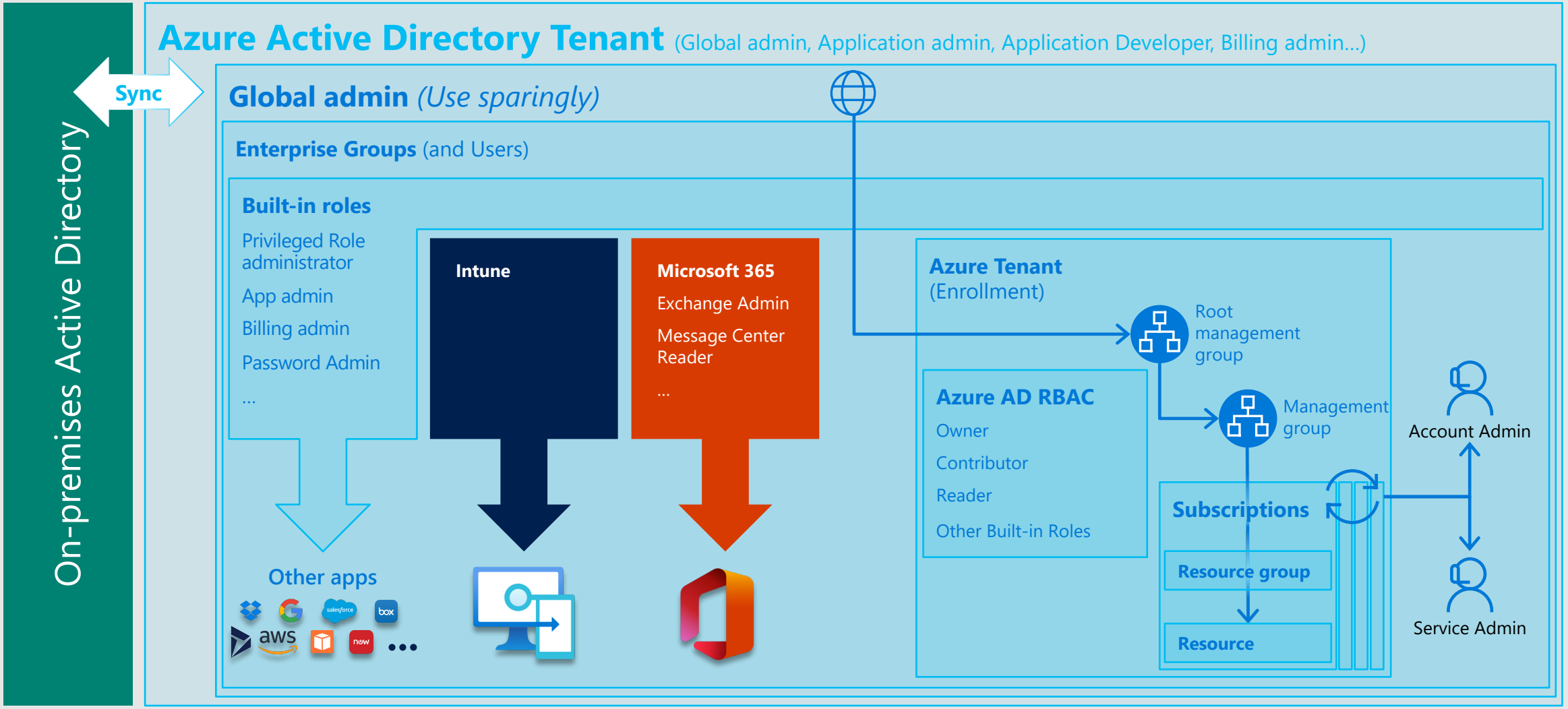
NOT, NOT & NOT 😁

Roles are in Azure AD but also in Azure but it's another story

Keep that **Roles or RBAC are provided from Azure AD security principals.**

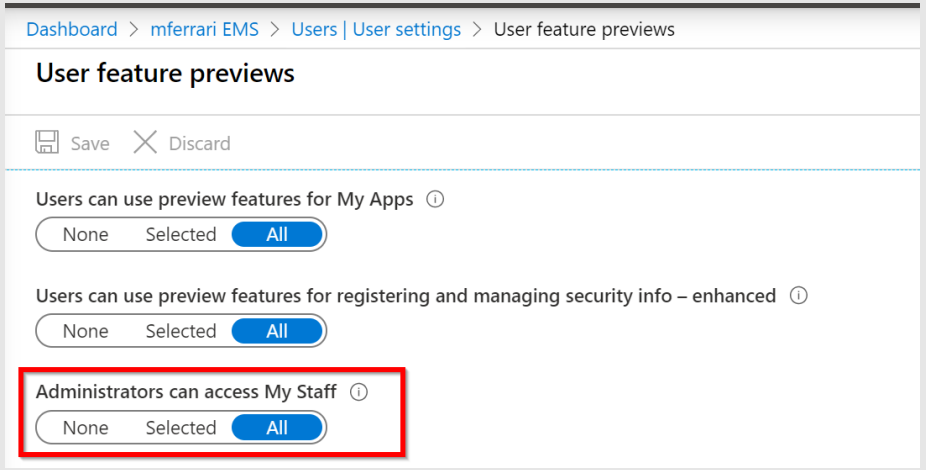


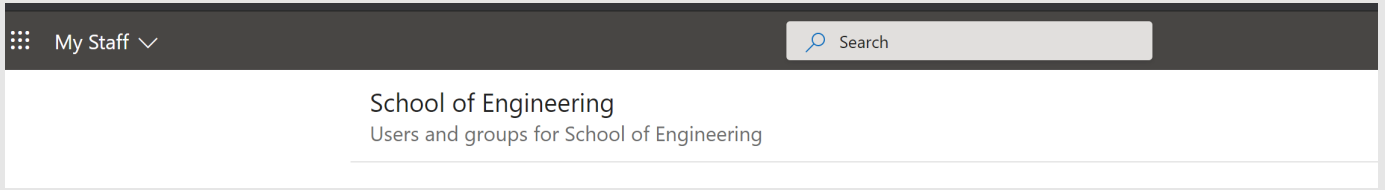
Understanding Azure RBAC & Azure AD Roles

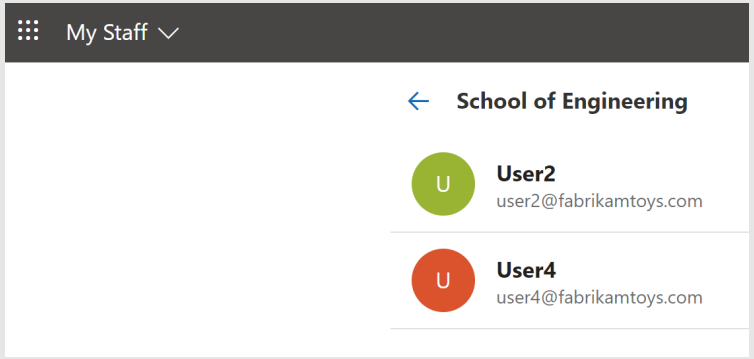


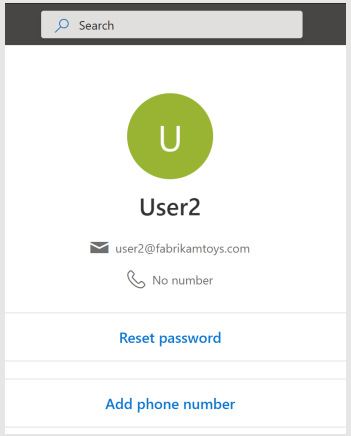
Another manner to delegate

Simple Credential Management experience designed to let Managers with delegated ‘scoped’ role membership manage credentials of end users.









- My Staff allows IT Admins to delegate scoped credential management to business leaders who know the needs of end users they manage.
- Delegating credential management reduces the load of Helpdesk, ad it allows for quicker turn around on getting credentials registered/updated.

Microsoft

List of abbreviations

AAD – Azure Active Directory

IAM – Identity and Access Management

IDaas – Identity as a Service