Cloud Identity Patterns and Strategies

Design enterprise cloud identity models with OAuth 2.0 and Azure Active Directory

Giuseppe Di Federico | Fabrizio Barcaroli

Chapter 1

Figures

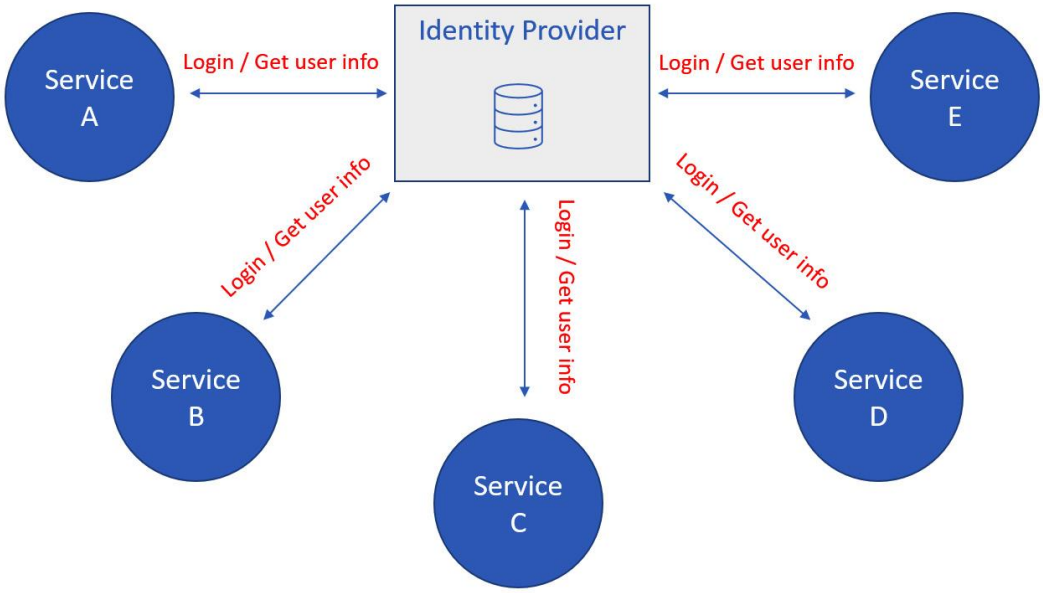


Figure 1.1 – IdP and service relationship

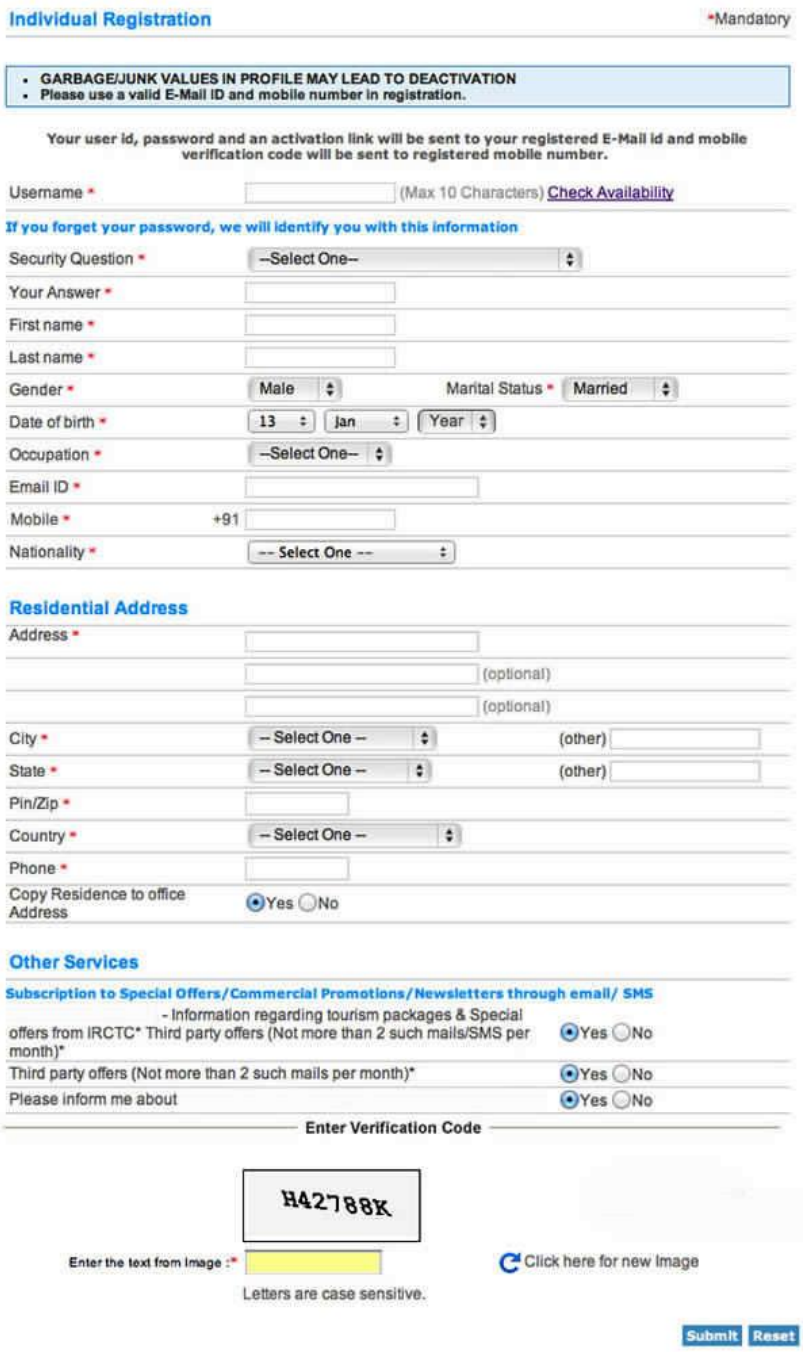


Figure 1.2 – Example of a long registration form, which is not so common nowadays

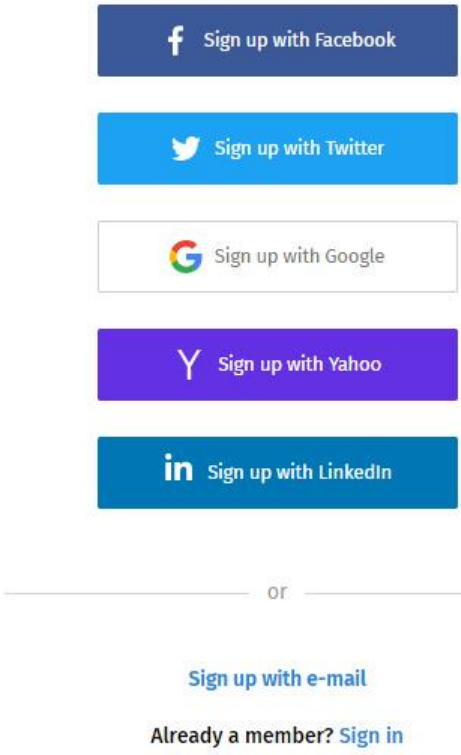


Figure 1.3 – Example of an external IdP signup

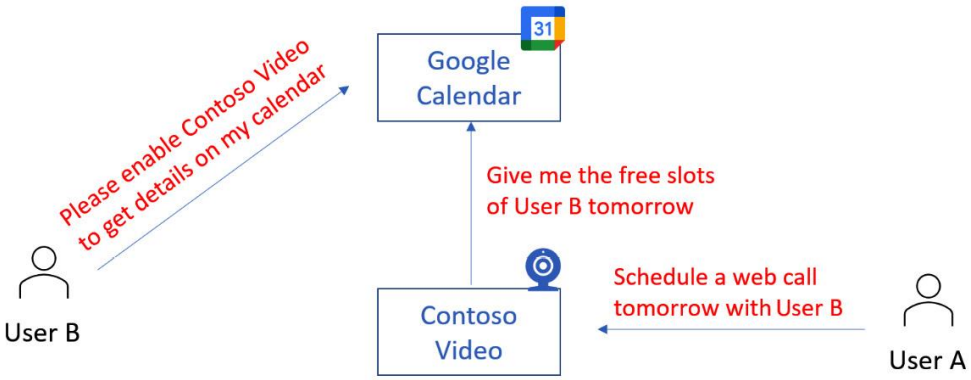


Figure 1.4 – Contoso Video user flow example

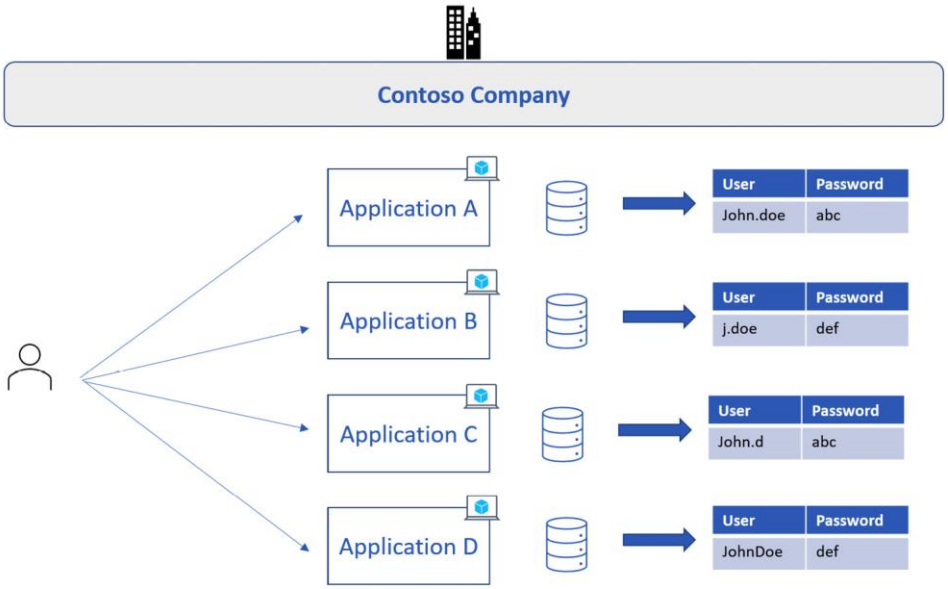


Figure 1.5 – Distributed identity problem example

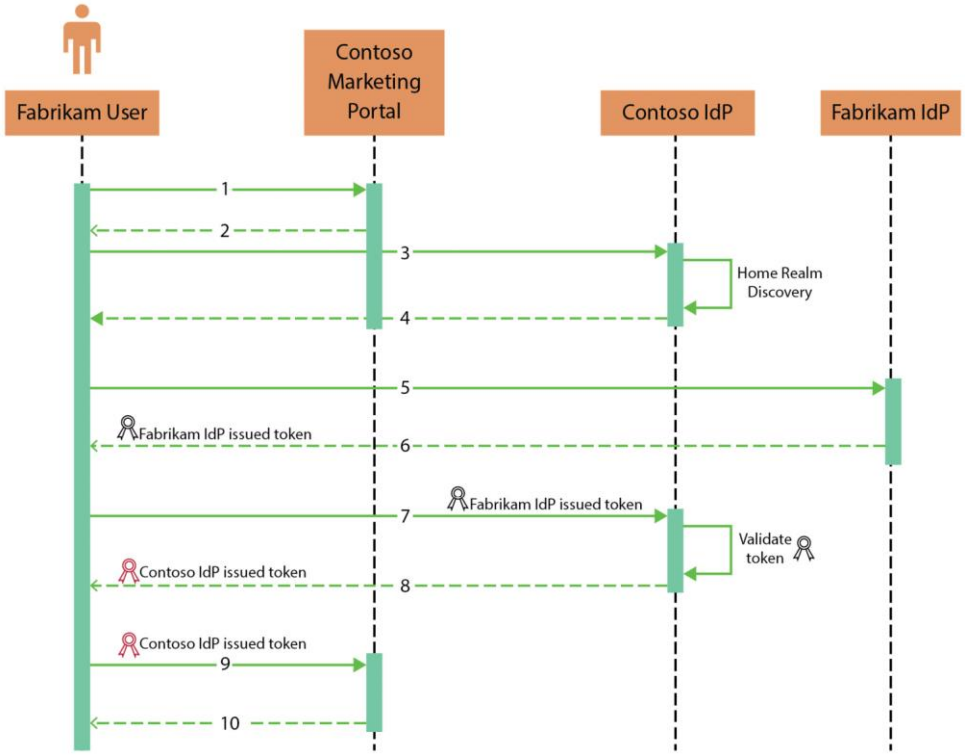


Figure 1.6 – User authentication flow with two IdPs

Code

Code 1.1: Example of an extract from a signed SAML response token

```
<?xml version="1.0" encoding="utf-16"?>
[..]
    <Issuer
```

```
xmlns="urn:oasis:names:tc:SAML:2.0:assertion">http://sts.katsuton.co
m/adfs/services/trust</Issuer>
    <samlp:Status>
        <samlp:StatusCode
Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
    </samlp:Status>
[..]
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
[..]
            <ds:SignatureMethod
Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" />
[..]
    <ds:SignatureValue>OUPJpFsnUODCK2h7T5SYMVhlWDnCBT6Qy
T9CcVnrjcWUPZTAaz2FNGEpPPhb/P9kW23cw5D1+fjhtAQurN/Du9uYfdkGtXcTPfcOO
VfuzgQT1d75HmYnbAtTvhsOrS8gvGCY6o
Jk3wsqNar3hrqLHDFxsszY41lZvOe2/Qax1SMpHeglQSbu6WOFe3sPdSiLY8rnWBE5Qu
bS85N1E+HNvjHqXS7Luwr
RDNK0InMM+LdPZw1YdOGUikgTbyIFKMR/eXR5UqbVrvmwv58XxT9C5p7FYPu3eKjWLD2
aGjCnJufFNfHiVGYrB8OU1FN1E/2sLNXnSuMyNnQJ5iWCQWP3vQ==</ds:SignatureV
alue>
[..]
        </ds:Signature>
<Subject>
<NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">fabarca@katsuton.com</NameID>
</Subject>
[..]
        <Conditions NotBefore="2021-06-28T09:26:39.720Z"
NotOnOrAfter="2021-06-28T09:27:39.720Z">
            <AudienceRestriction>
    <Audience>urn:microsoft:adfs:claimsxray</Audience>
            </AudienceRestriction>
        </Conditions>
        <AttributeStatement>
            <Attribute
Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name">
      <AttributeValue>kadmin</AttributeValue>
            </Attribute>
            </AttributeStatement>
[..]
</samlp:Response>
```

Chapter 2

Figures

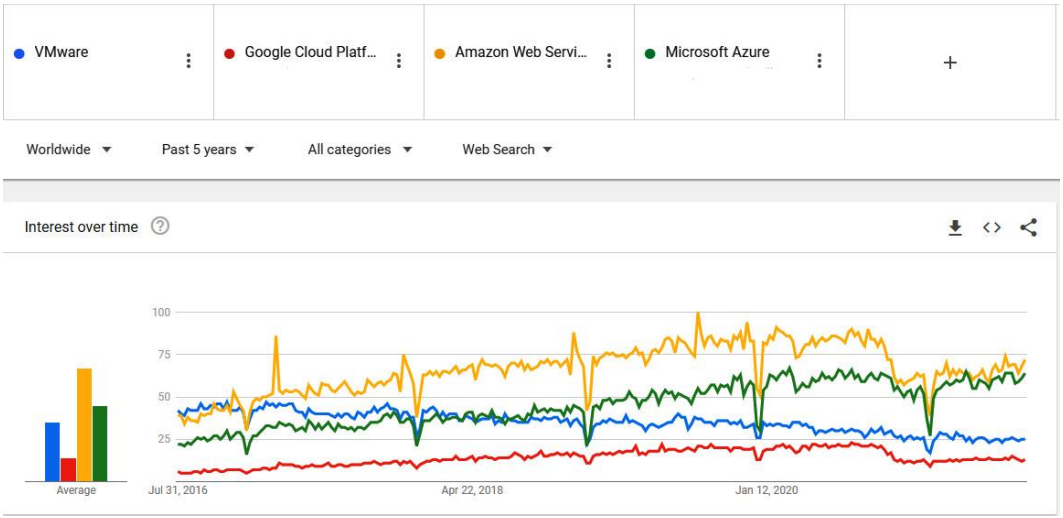


Figure 2.1 – Evolution of cloud trends over time versus the most famous on-premises virtualization according to Google trends

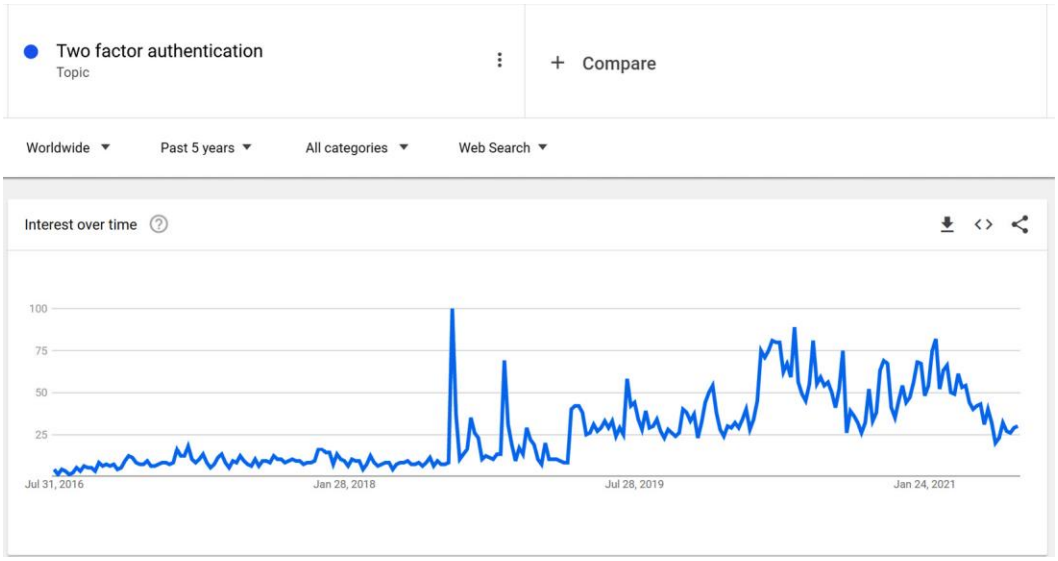


Figure 2.2 – The search trends for two-factor authentication on Google in the last 5 years

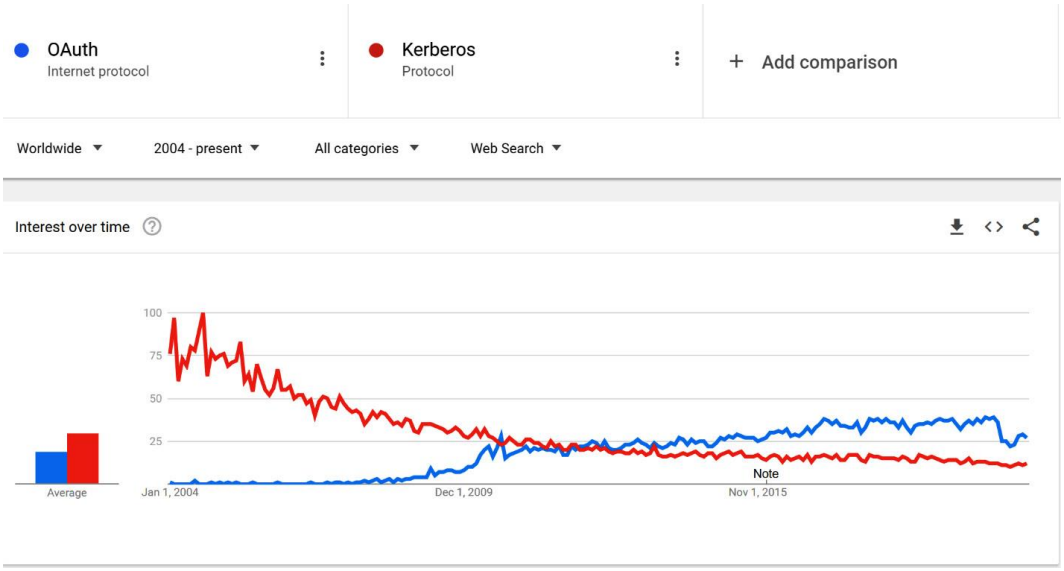


Figure 2.3 – Trend comparison between a historical enterprise protocol (Kerberos) versus Oauth

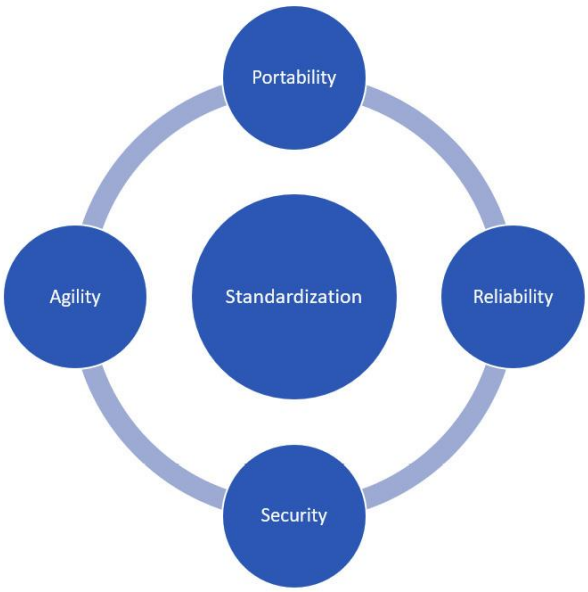


Figure 2.4 – The pillars of a cloud company

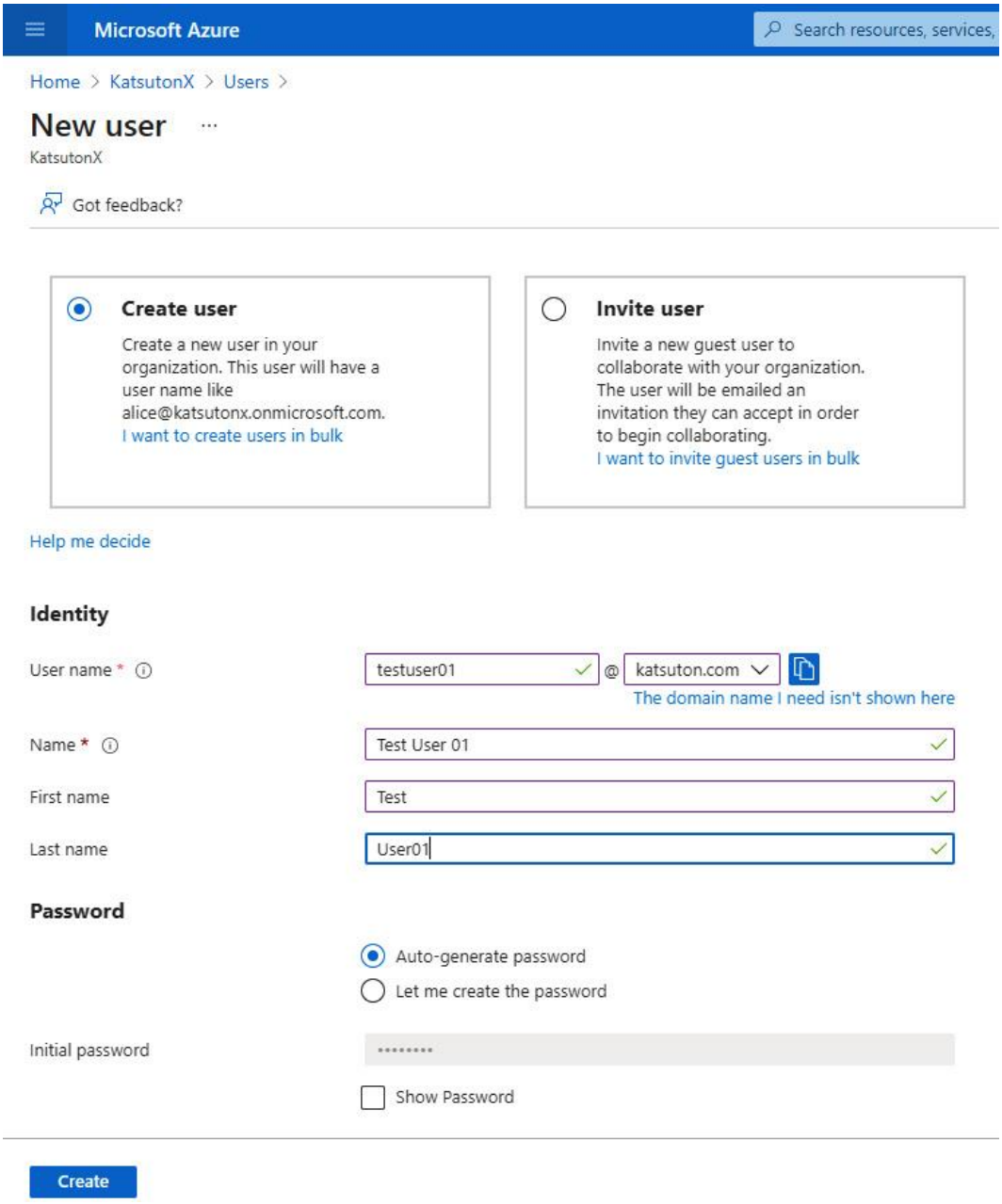


Figure 2.5 – Azure AD user creation example

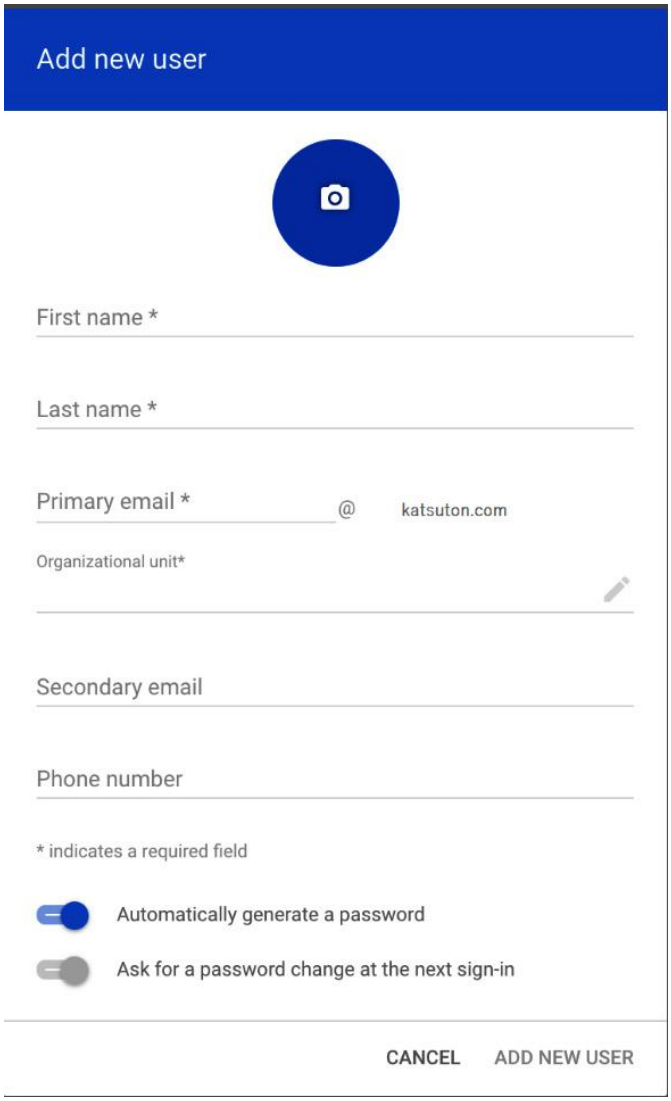


Figure 2.6 – Google user creation example

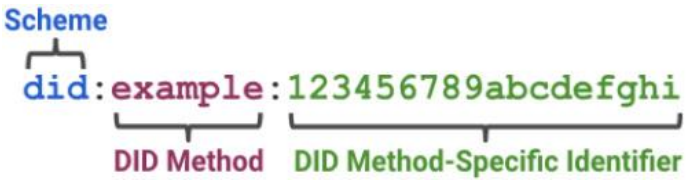


Figure 2.7 – An example of DID

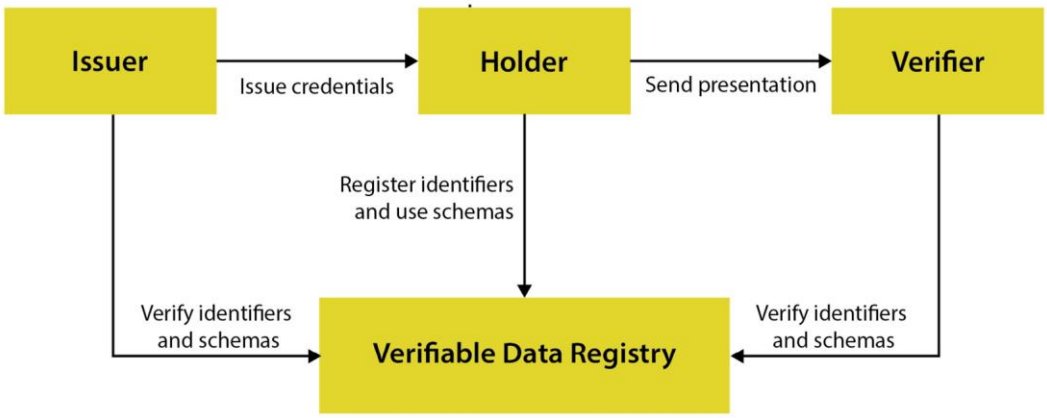


Figure 2.8 – Verifiable credentials example

Links

OpenID Connect Federation (https://openid.net/specs/openid-connect-federation-1_0.html)

ION Zero-knowledge proof protocol (https://www.blockchain-council.org/blockchain/zero-knowledge-proof-protocol/)

Decentralized Identifiers (**DID-core**) (https://www.w3.org/TR/did-core/)

Preparing for a more private web | Session: https://www.youtube.com/watch?v=1g2uQfP1Q3U&t=138s

Business Directory: (https://businessdictionary.info/)

for those who are still interested in creating their own solution, there are several GitHub repositories that may help you with that, such as https://github.com/IdentityServer/IdentityServer4 , now turned into a commercial solution).

Chapter 4

Figures

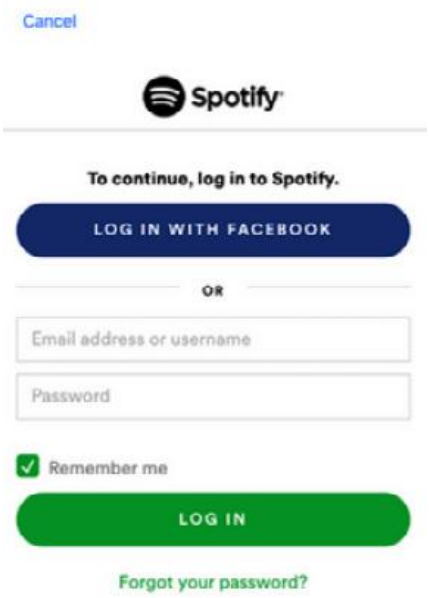


Figure 3.1 – Sample of OAuth login

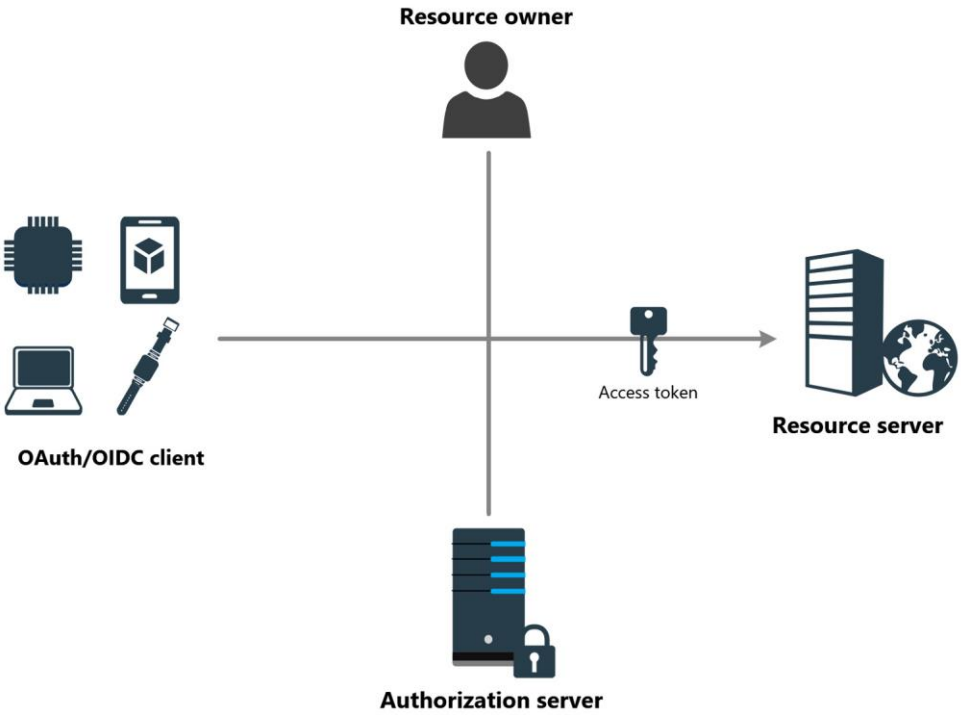


Figure 3.2 – OAuth/OIDC parties

Code

Code 3.1

```
{
 "typ": "JWT",
 "alg": "RS256",
 "kid": "JRX9C2vPdG2-3YDi14buhEVvO_O9bFwzz-pMBxtDh18"
}.{
 "exp": 1633425451,
 "nbf": 1633421851,
 "ver": "1.0",
 "iss": "https://myauthzserver.issuer.com/",
 "sub": "application_or_user_unique_id",
 "aud": "audience_unique_id",
 "acr": "signin",
 "nonce": "defaultNonce",
 "iat": 1633421851,
 "auth_time": 1633421851,
 "name": "unknown"
}.[Signature]
```

Links

Welcome to OpenID Connect: (https://openid.net/connect/)

Authentication flows and application scenarios: https://docs.microsoft.com/en-us/azure/active-directory/develop/authentication-flows-app-scenarios.

More information about the OAuth 2.0/OIDC authorization server metadata can be found in the OAuth 2.0 specification, under *Obtaining Authorization Server Metadata* at the following URL: https://datatracker.ietf.org/doc/html/rfc8414

Access tokens: https://datatracker.ietf.org/doc/html/rfc6749

Chapter 4

Figures

Protocol /Flow	Authorization Code Grant	Implicit	Client Credentials	Resource Owner Password Credentials	On-Behalf-Of	Hybrid
OAuth 2.0	YES	YES	YES	YES	YES	YES
OIDC	YES	YES	NO	NO	NO	YES

Figure 4.1 – OIDC/OAuth 2.0 flow support summary

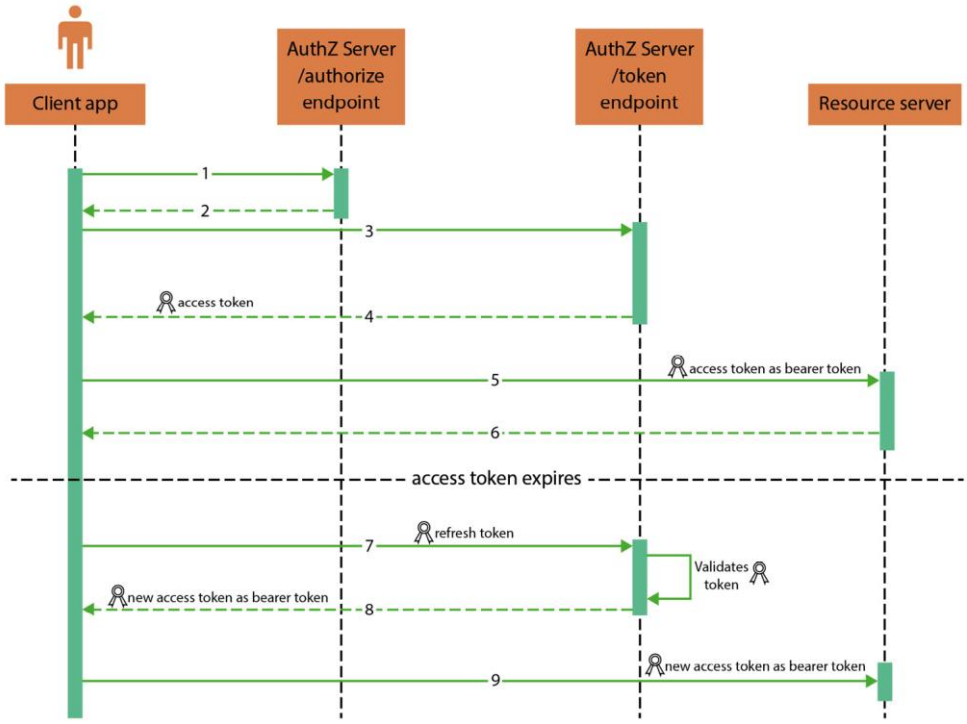


Figure 4.2 – Authorization code grant flow

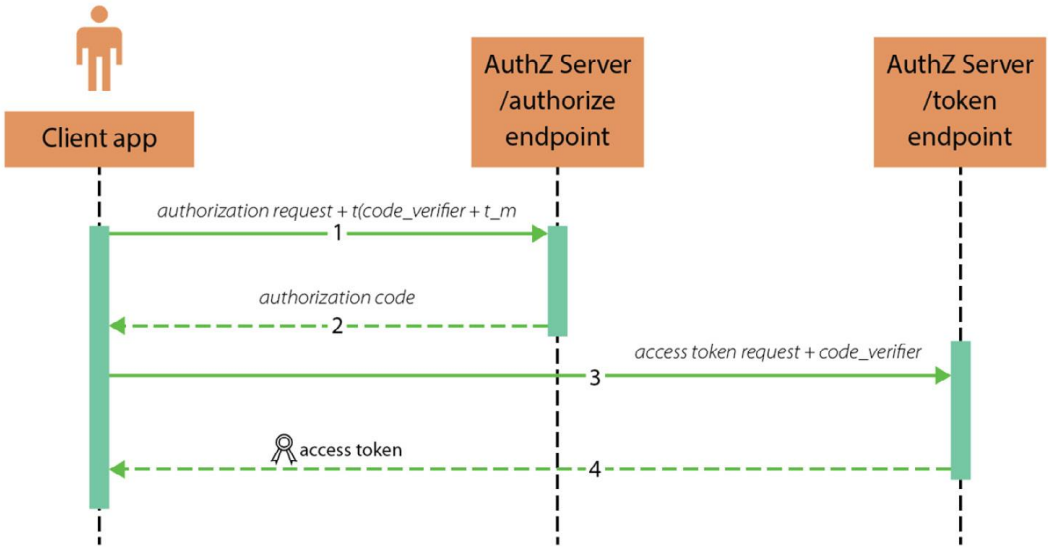


Figure 4.3 – PKCE

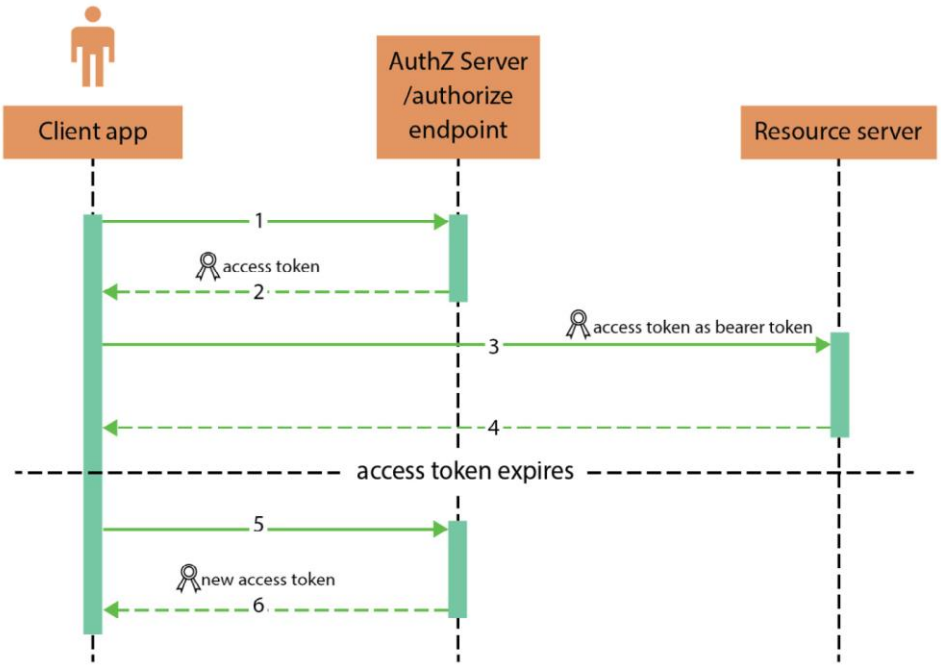


Figure 4.4 – Implicit grant flow

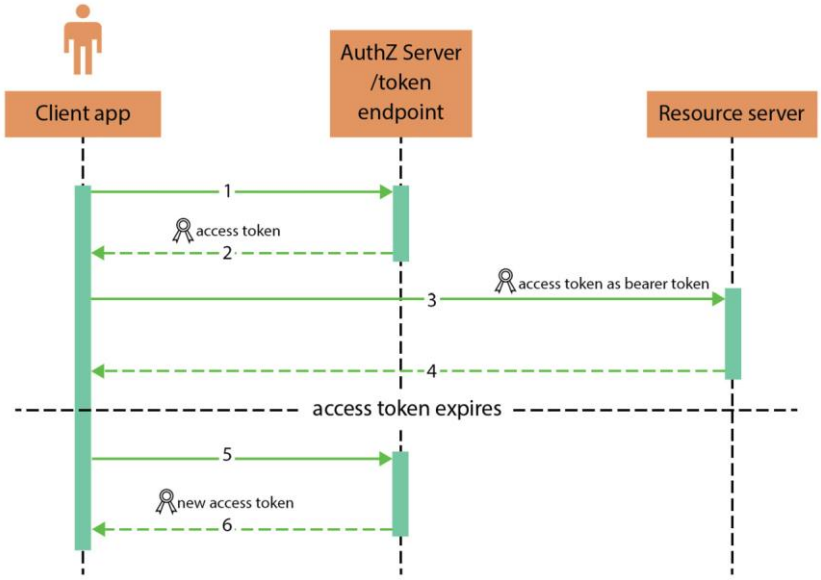


Figure 4.5 – Client credentials grant flow

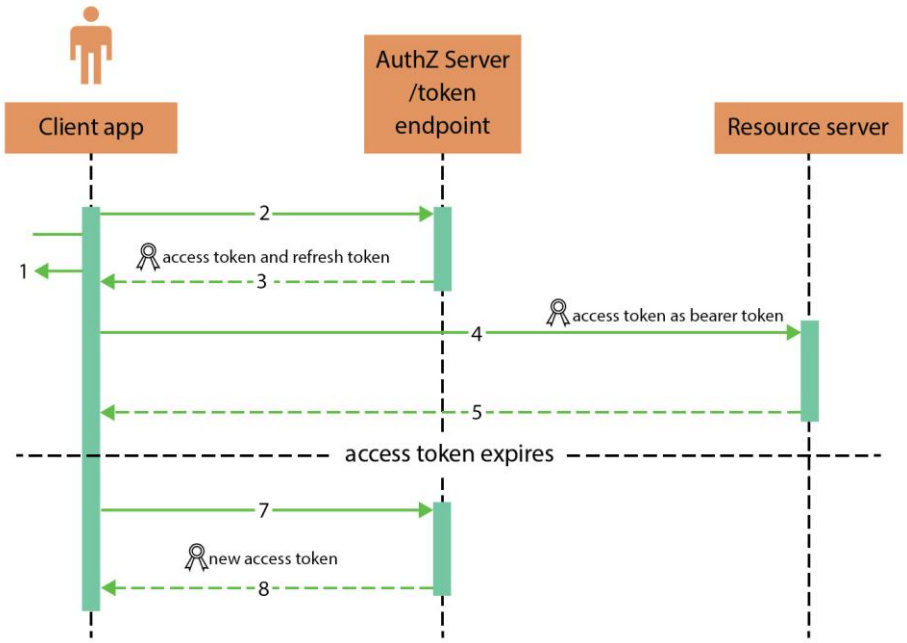


Figure 4.6 – ROPC grant flow

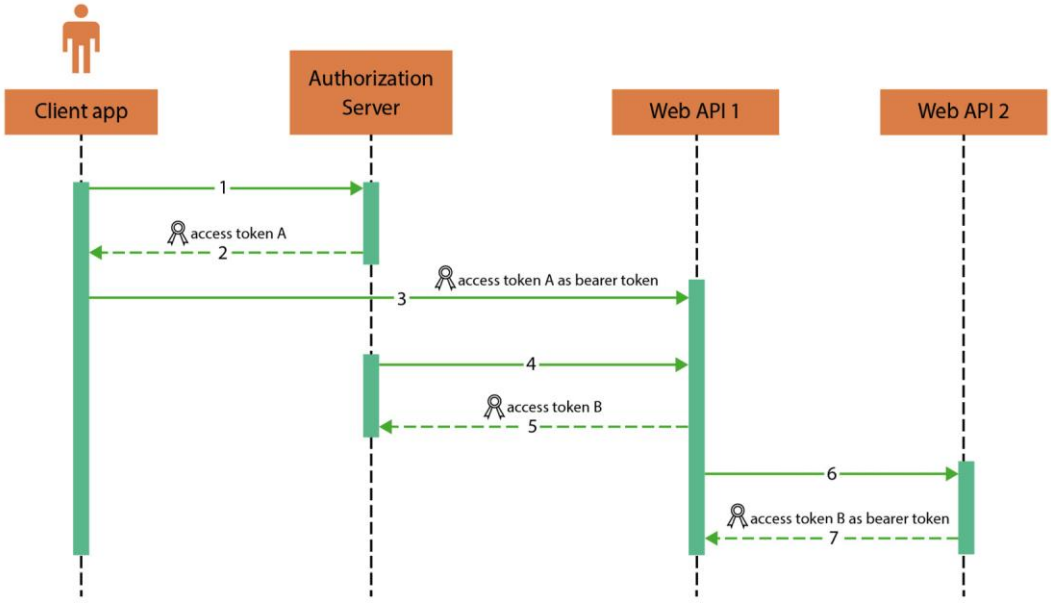


Figure 4.7 – OBO flow

Codes

Code 4.1: The authorization code grant flow

```
GET /authorize?
response_type=code
&client_id=s6BhdRkqt3
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
&scope=openid%20resource_server_id
&nonce=n-0S6_WzA2Mj
&state=af0ifjsldkj HTTP/1.1
Host: autzserver.example.com
```

Code 4.2: The authorization code grant flow

```
Host: authzserver.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
  &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

Code 4.3: The implicit grant flow

```
GET /authorize?
```

```
response_type=token    (or id_token)
&client_id=s6BhdRkqt3
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
&scope=openid%20resource_server_id
&nonce=n-0S6_WzA2Mj
&state=af0ifjsldkj HTTP/1.1
Host: autzserver.example.com
```

Code 4.4: The client credentials grant flow

```
GET /token?
grant_type=client_credentials
&client_id=s6BhdRkqt3
&scope=resource_server_id
&client_secret=uayaskiR$£QDcfa
Host: authzserver.example.com
```

Code 4.5: The ROPC grant flow

```
POST /token?
grant_type=PASSWORD
&client_id=s6BhdRkqt3
&scope=resource_server_id%20offline_access
&username=userid1
&password=84ru2hkajhf
Host: authzserver.example.com
```

Code 4.6: Hybrid Flows

```
GET /authorize?
 response_type=id_token%20token
 &client_id=s6BhdRkqt3
 &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
 &state=bfansly HTTP/1.1
Host: authzserver.example.com
```

Chapter 5

Figures

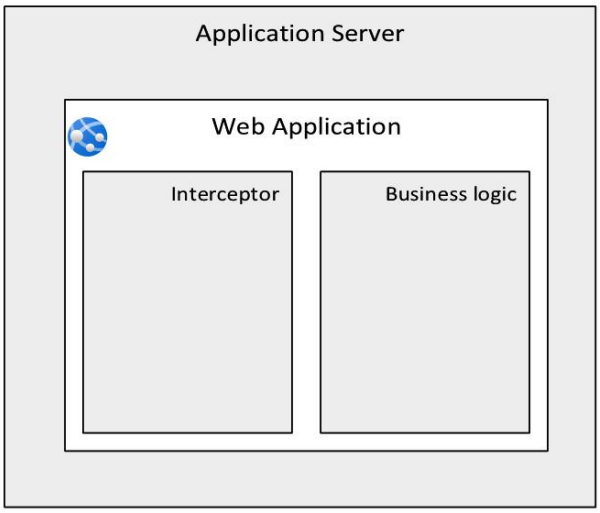


Figure 5.1 – Web application layers

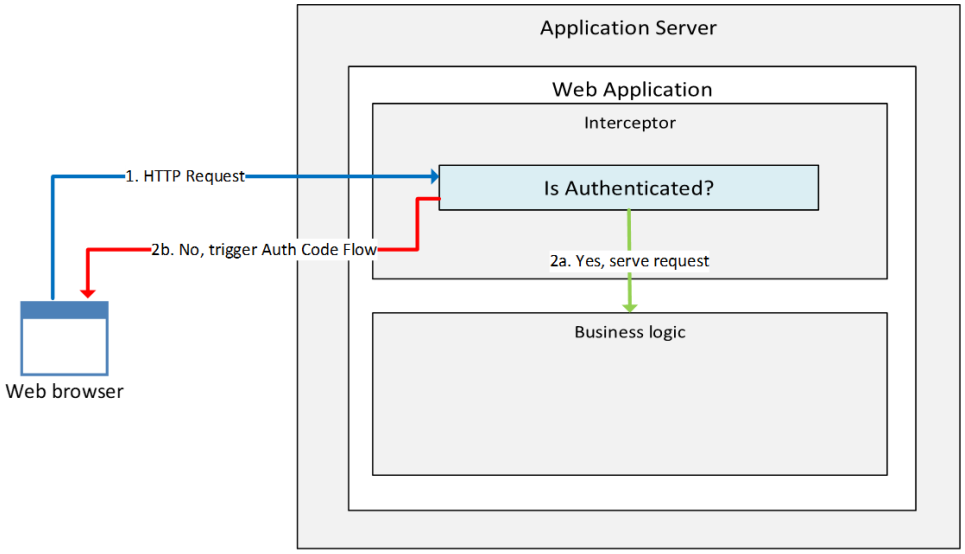


Figure 5.2: Figure 5.2 – Sample application logic for an AuthZ code grant flow

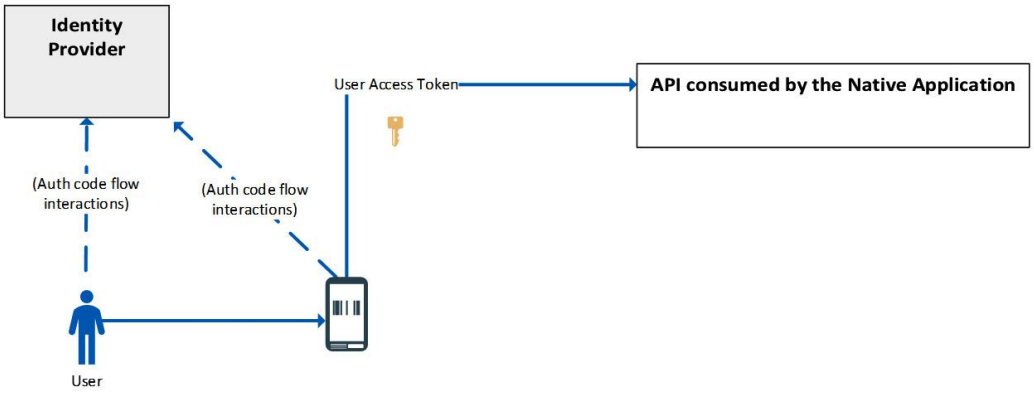


Figure 5.3 – Native app – application authorization pattern

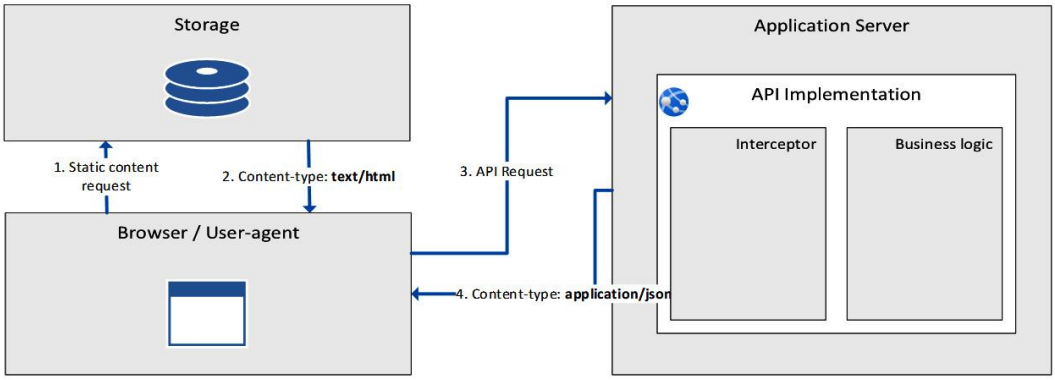


Figure 5.4 – SPA – logical diagram

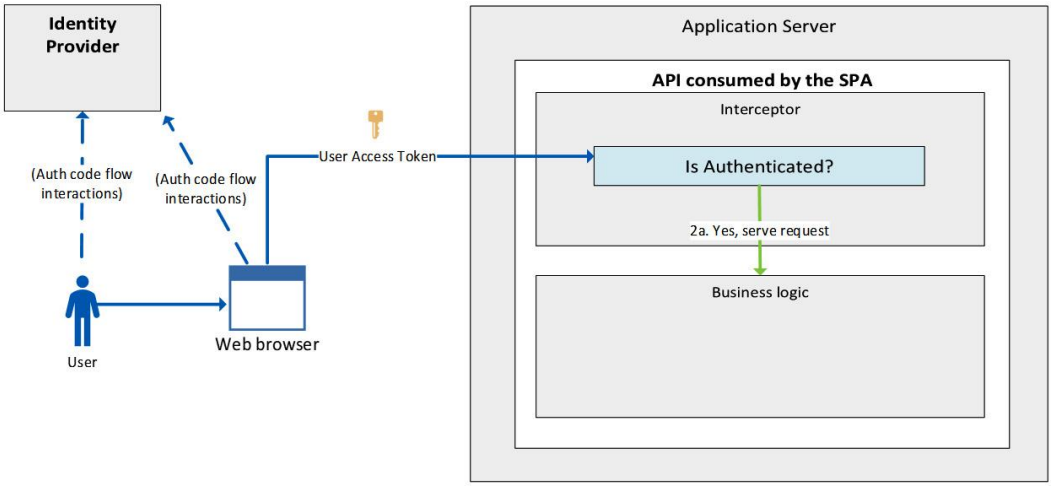


Figure 5.5 – Single-page application pattern – a monolith design

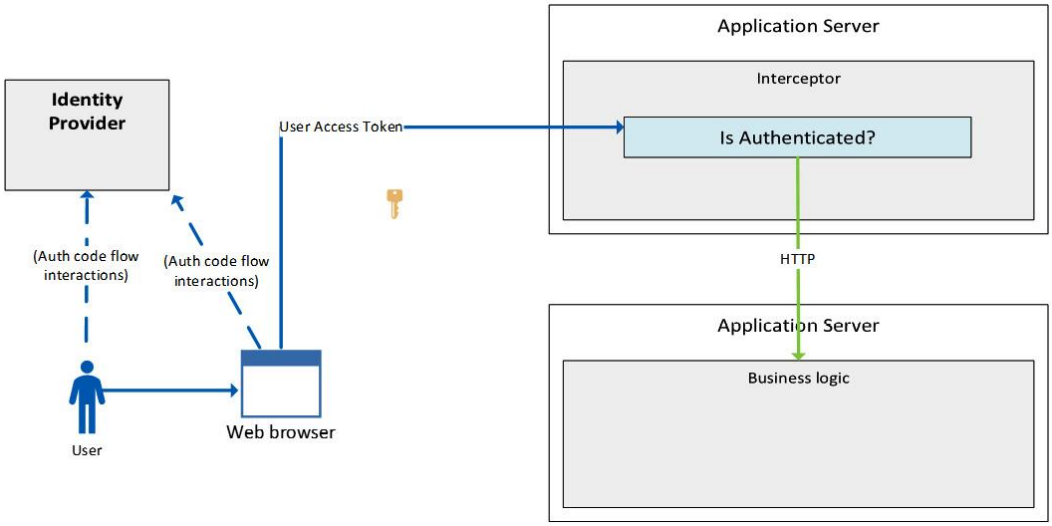


Figure 5.6 – Decoupled architecture

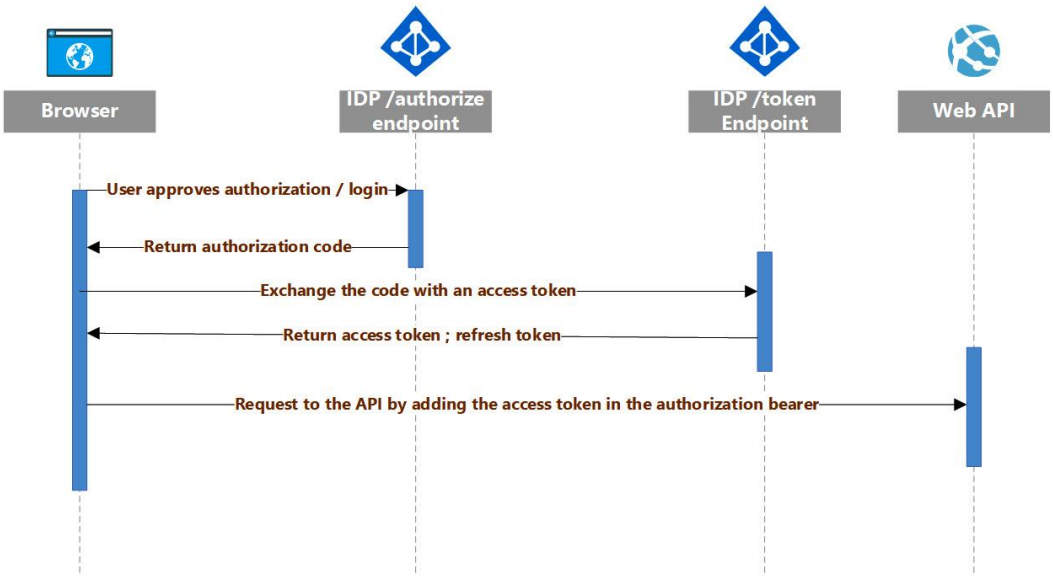


Figure 5.7 – SPA authorization sequence (the Authorization Code Grant flow)

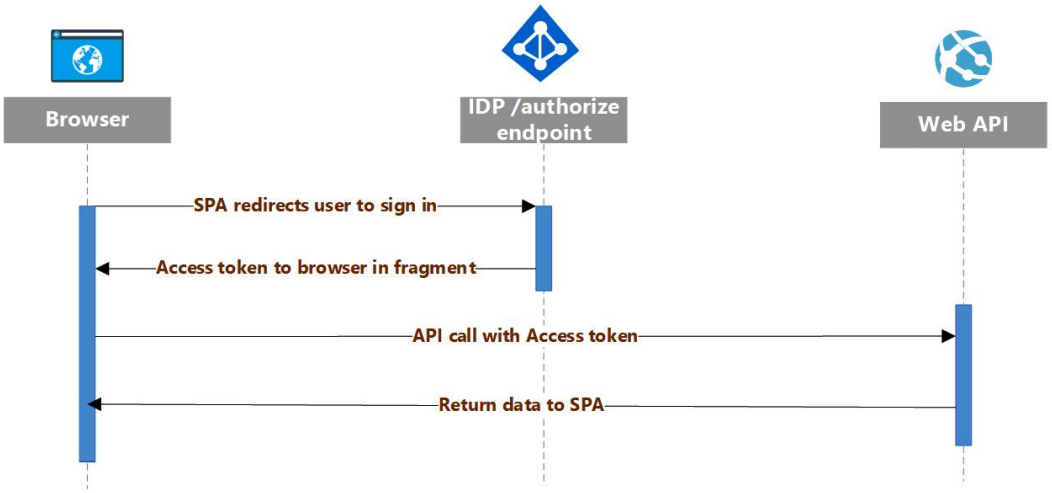


Figure 5.8 – SPA authorization sequence (the implicit flow)

Table

Step	Description	Sample code/Sample HTTP request
1.	The static content served to the web browser will propose the option to the user to authorize the application to call the API on their behalf.	**Connect Your Account**
2.	The user is taken to the auth server and can log in or approve the OAuth request if already logged in. Once the operation completes, the authorization server will redirect the user to the redirect URI specified in the app registration, along with a code that will be used later by the browser to grab the access token.	https://contoso.com/cb?code=sampleCode123
3.	The app makes a **POST** request to the service's token endpoint in order to exchange the authorization code for an access token.	**POST /oauth/token HTTP/1.1** **Host: authorization-server.com** **grant_type=code** **&code=sampleCode123** **&redirect_uri=https://contoso.com/cb** **&client_id=sampleClient**
4.	At this point in time, the SPA has the access token that will be included in the host headers of the calls to be authorized by the API.	**Authorization: Bearer <access_token>**

Table 5.1 – OAuth steps for a SPA (example)

Codes

Code 5.1: User authentication only pattern

```
GET /authorize?
response_type=code
&client_id=WebApp1-Id
```

```
&redirect_uri=https%3A%2F%2Fwebapp1.example.com%2Fauth
&scope=openid%20profile
&nonce=n-0S6_WzA2Mj
&state=af0ifjsldkj HTTP/1.1
Host: idp1.example.com
```

Code 5.2: User authentication only pattern

```
POST /token HTTP/1.1
Host: idp1.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic WebApp1-Secret
grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
  &redirect_uri=https%3A%2F%2Fwebapp1.example.org%2Fauth
```

Code 5.3: Application authorization pattern

```
GET /authorize?
response_type=code
&client_id=NativeApp1-Id
&redirect_uri=msalc38ukiaghksg73ldsg%3A%2F%2Fauth
&scope=NativeApp1-Id%3Aread%20NativeApp1-Id%3Awrite
&nonce=n-0S6_WzA2Mj
&state=af0ifjsldkj HTTP/1.1
Host: idp1.example.com
```

Code 5.4: Application authorization pattern

```
POST /token HTTP/1.1
Host: idp1.example.com
Content-Type: application/x-www-form-urlencoded
grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
  &redirect_uri=msalc38ukiaghksg73ldsg%3A%2F%2Fauth
```

Links

More information about the sidecar pattern can be found here: https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar

Chapter 6

Table

API	Description	Example
Application frontend API	An HTTP endpoint that belongs to the application and is designed to be consumed by the application's user	**Single-Page Application** (**SPA**)
Application automation API	A publicly exposed HTTP endpoint that belongs to the application and is designed to be consumed by an automation service in a controlled way	Automatic processes need to query the application
Application backend API	An internally exposed HTTP endpoint that belongs to the application and is designed to be consumed by other applications' components	**Service Oriented Architecture** (**SOA**), modular applications, microservice applications
Internal reusable API	An HTTP endpoint that does not belong to any specific application and provides a service that can be consumed	A company's CRM that needs to provide common information that can be consumed by different applications
Partner API	An HTTP endpoint that exposes internal services to external partners and companies	A Visa credit card company that exposes services to be consumed by partners to manage transactions

Table 6.1 – Types of API

Pillar to structure the company's APIs	Description
Discovery	As reported previously, regardless of its physical location, for an API to be adopted, it is important to keep an internal registry within the company that can help the developers to find an API they may need to reuse to facilitate their development. As we're going to see in upcoming chapters, certain technologies have this capability out of the box.
Monitoring	An API cannot be managed efficiently without live information that can report how it is being used and expose performance metrics. There are many tools that can help to define the monitoring pillar, and plenty of telemetry technology the API can take advantage of. As for the other pillars, it is extremely important to have a good level of consistency across the enterprise to simplify the maintenance and troubleshooting by the ops team, which should not switch from one tool to another according to the target API they need to monitor. It is important to understand that this

Pillar to structure the company's APIs	Description
	pillar is usually connected with the development pillar because a technology choice can constrain what monitoring tools can be adopted.
Security	This is a hard topic to create a standard for. The reason is that each API can have different security requirements. For example, one API can expose sensitive internal information, while another API may be intended for public use to access information that isn't sensitive. The advice here is to create different security tiers to enable the architect that is planning the API to choose the proper security tier according to the API target usage and requirements.
Authentication	This pillar needs to outline what an API consumer needs to do to authenticate against the API. The most important aspects this pillar needs to clarify are as follows: It needs to report whether the authentication needs to be based on OAuth and whether application registration is required It needs to report whether the authentication logic needs to be implemented within the business logic or offloaded to an external component, such as the API management layer of a service mesh Ideally, across the organization, it is recommended to adopt, if possible, a single IdP that needs to be defined by this pillar. This can help to simplify the processes and the overhead reported in the *The multiple IdP dilemma* section.
Deployment	This is usually connected to the DevOps practices adopted by the company, and it is recommended to be consistent with them. It is important that the whole deployment process is immutable and automatic to deploy and promote the API across different environments. If the API needs to be exposed externally, it is recommended the automation takes care of not only the deployment but also its exposure, and if OAuth is involved, the registration across the specific IdP.
Testing	This relates to what needs to be tested on an API and how it is supposed to be tested. Generally speaking, each API needs to be developed with unit testing. If the API is part of a microservices application or is in the general part of a serverless flow or an end-to-end flow, it is important to also define the integration tests needed to ensure it will work. Performance testing is an important aspect as well, and it is important to outline whether a distributed load test is needed and if so, what the benchmark is. Testing an API also has authentication implications because the test needs to mimic the caller and replicate the authentication flow of the caller, whether an OAuth client credential flow or a different protocol or flow.
Development	The development pillar is maybe the widest one in terms of what decisions need to be made. It defines, for example, which frameworks and

Pillar to structure the company's APIs	Description
	technologies the company intends to support. It is important to note that these choices may have specific implications relating to monitoring, security, and testing, as they can, in turn, be based on technologies (such as libraries and SDKs) that need to be compatible with this pillar. Another important factor to decide how to define this pillar is the development cost. There are specific technologies that are easy to adopt for API development; other technologies where, for example, it is hard to find skilled developers; and others that are widely used but may require further overhead in API development. All of these aspects need to be carefully considered. It is important to outline that an API can be written in any technology without impacting the consumer, who just needs to access the HTTP endpoint. From a maintenance perspective, a company cannot have all the APIs developed differently because it would create issues in the future to find proper staff for maintenance purposes.
Design	The external design (the interface) should be well-defined. This can define the degree of re-usability and enable the client to properly consume the API. As part of this pillar, the team needs to define what message patterns and protocols the API will adopt (such as REST or GraphQL).
Strategy	This is the tactical goal of the API, as in, what value the API will bring to the organization and how much of a strategic impact is measured. It is paramount to define its life cycle and to be able to evaluate in the long term when it is time for the API to be removed or redeveloped.
Documentation	The developer should be able to start using the API easily with specific guidelines. (For example, how to reach the API on the network. Is it on the internet or the intranet? Is it hosted on a cloud service and if so, does it require any further security enablement, such as firewall exceptions?)

Table 6.2 – Decision pillars

Pillar	Description
Discovery	The discovery or service discovery has duties similar to the ones of the **Domain Name Service** (**DNS**). In other words, when a service or API needs to communicate with another service within the cluster, it needs to locate it. The orchestration platform provides the *discovery service* to enable each service to locate and reach the target API/service. This concept is even more important when there are multiple physical clusters in place. The goal of discovery is to abstract from the physical layer and let the API management layer route the traffic to the target service regardless of its physical location. In other words, the API management layer always knows how to route the traffic to reach a specific service in a specific cluster, and the service is located in the node of the cluster.

Pillar	Description
Load balance	It is important for a company, especially when the logical view represented in *Figure 6.8* is made up of multiple physical clusters, to control the traffic among the API by implementing rules and performing advanced operations, such as blue-green deployment.
Traceability	The ability to trace requests, store them, and reproduce them enables companies to have an important asset that they can reproduce easily with the goal, and analyzing it, to either improve the quality, spot bugs, or boost their troubleshooting capabilities.
Observability	Having real-time telemetry for each deployment is paramount to enable a company to scale and introduce more APIs without losing its effectiveness to troubleshoot. Generally speaking, services, nodes, and clusters need to be consistently monitored at any point in time to enable the API to grow and keep the management overhead small at the same time. Unlike traceability, usually, observability is intended to be a high-level view of the service that is above the network stack.
Encryption	The messages exchanged between the services within the containerization platform should be encrypted and decrypted in a way that should not impact the logic of the services or APIs. Offloading the encryption to the containerization platform is a great benefit.
AuthNZ	Authentication and authorization, the ability to enable one service to communicate with another, can be centrally managed by the containerization platform.

Table 6.3 – The service mesh pillars

Figures

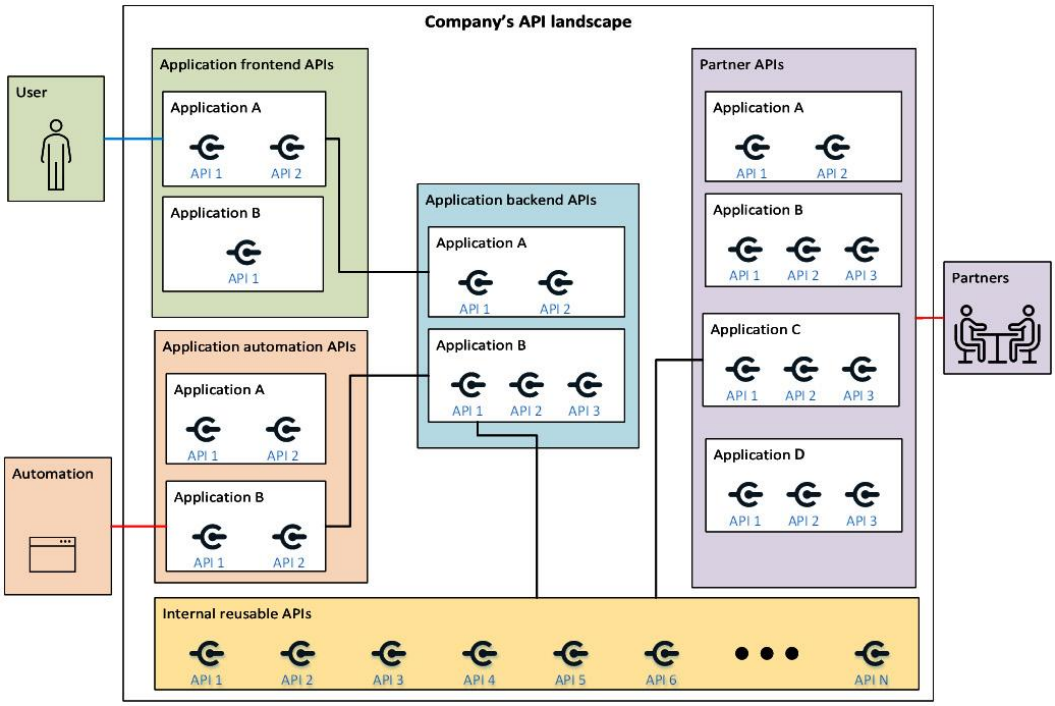


Figure 6.1 – A typical company's API landscape

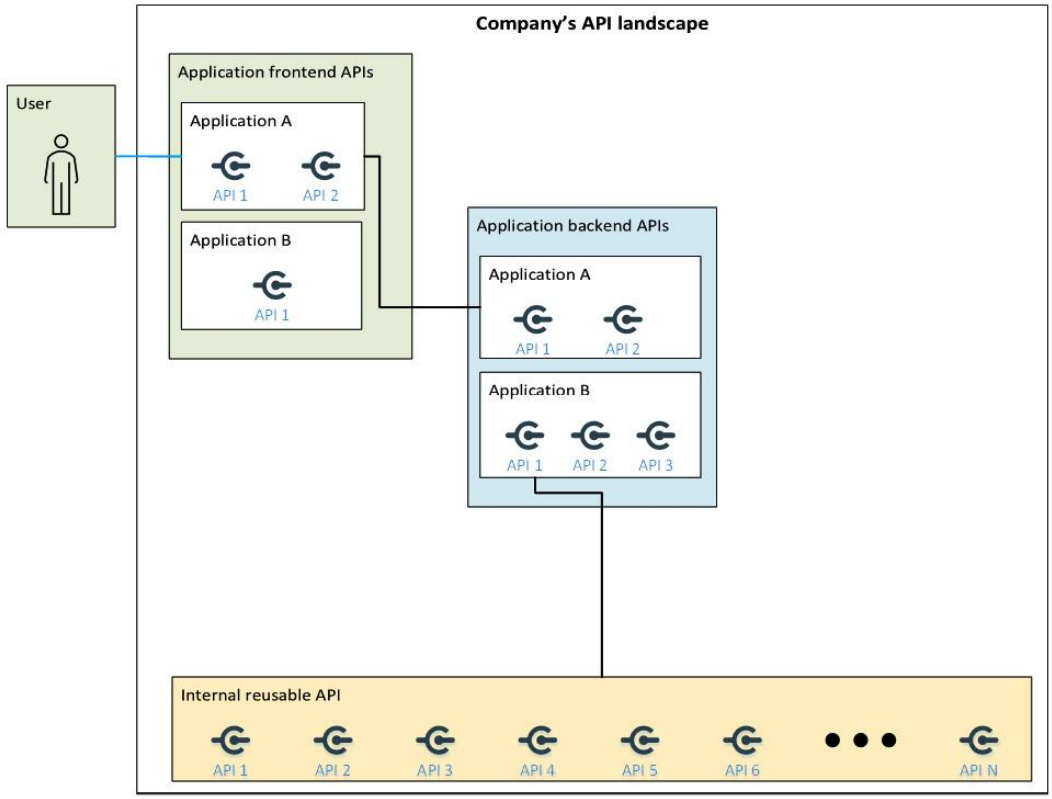


Figure 6.2 – Zooming in on the application frontend API

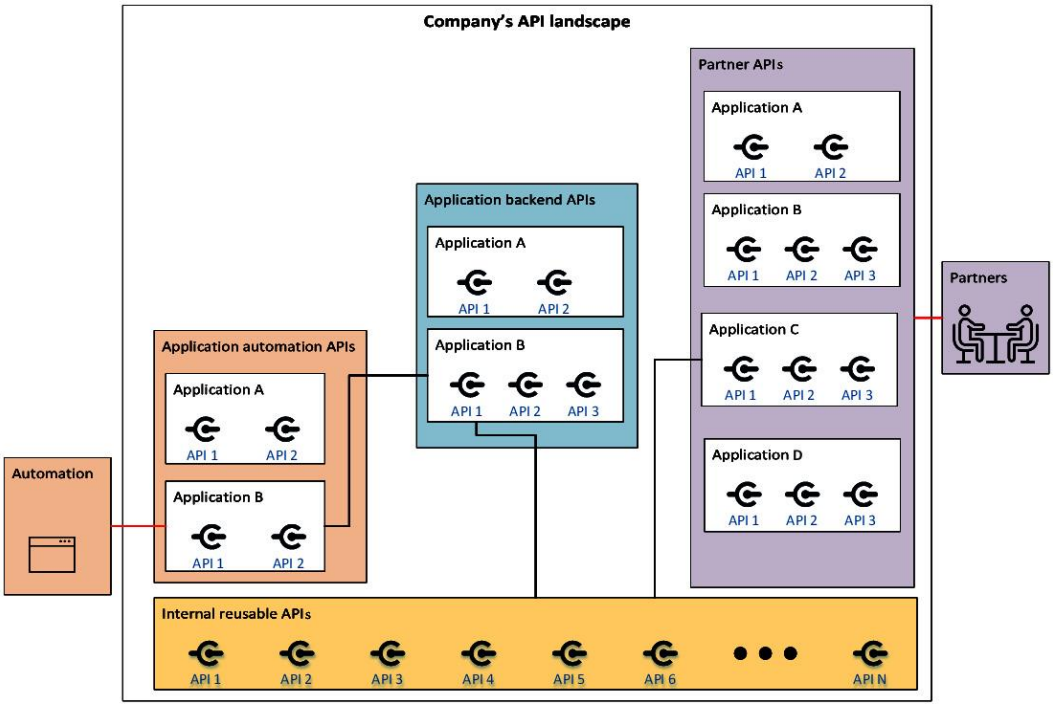


Figure 6.3 – Zooming in on the application automation APIs

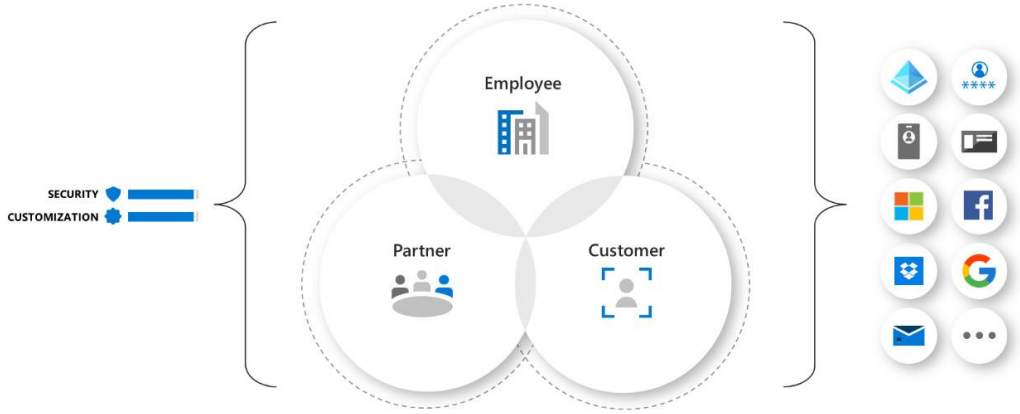


Figure 6.4 – Example of user categories

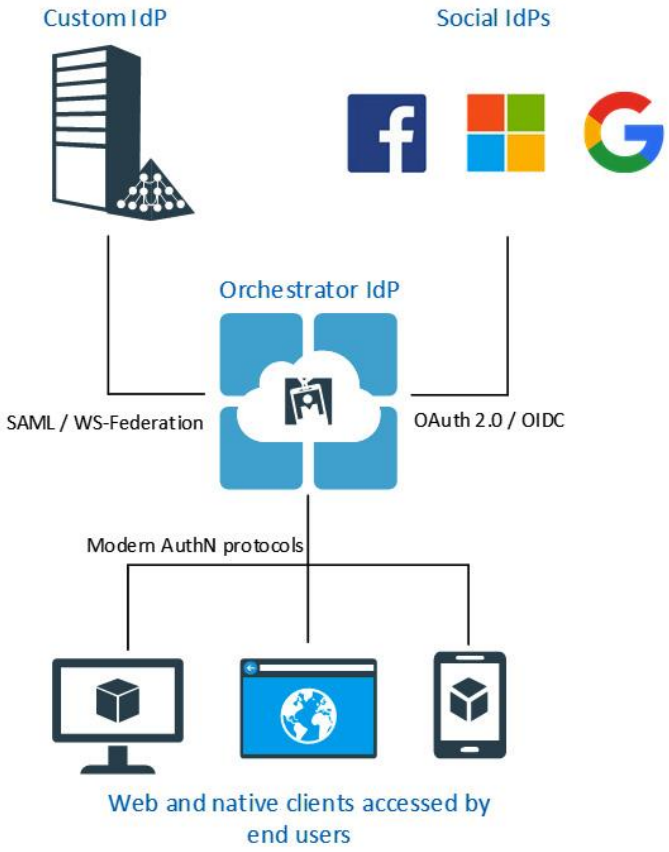


Figure 6.5 – Example of an IdP federating with multiple IdPs

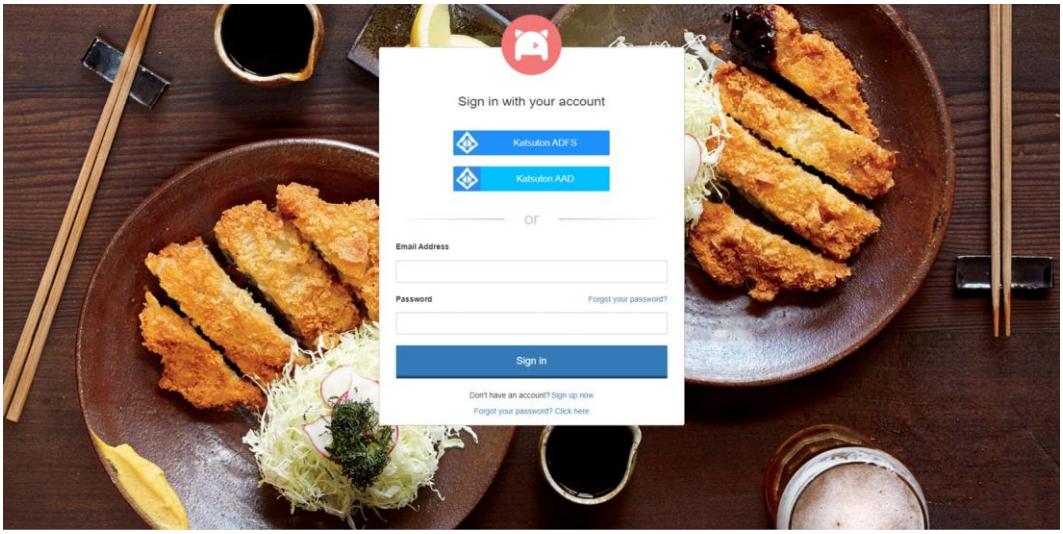


Figure 6.6 – Example of a login page of an IdP federated with other IdPs

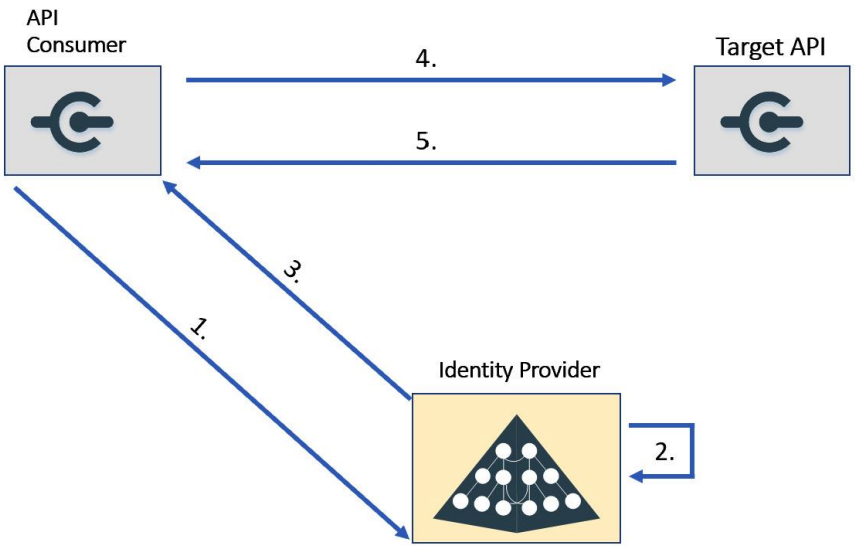


Figure 6.7 – API authentication with OAuth

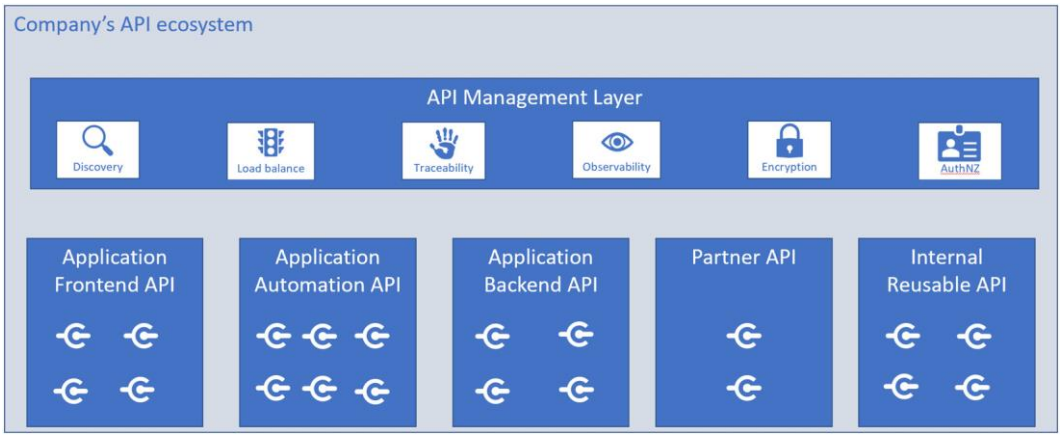


Figure 6.8 – Logical API grouping in a single-pane-of-glass fashion

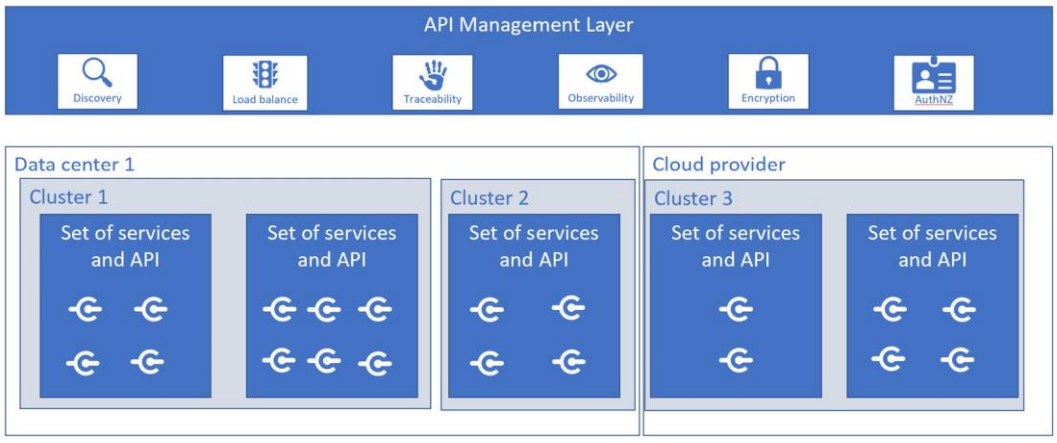


Figure 6.9 – Logical and physical view of a service mesh

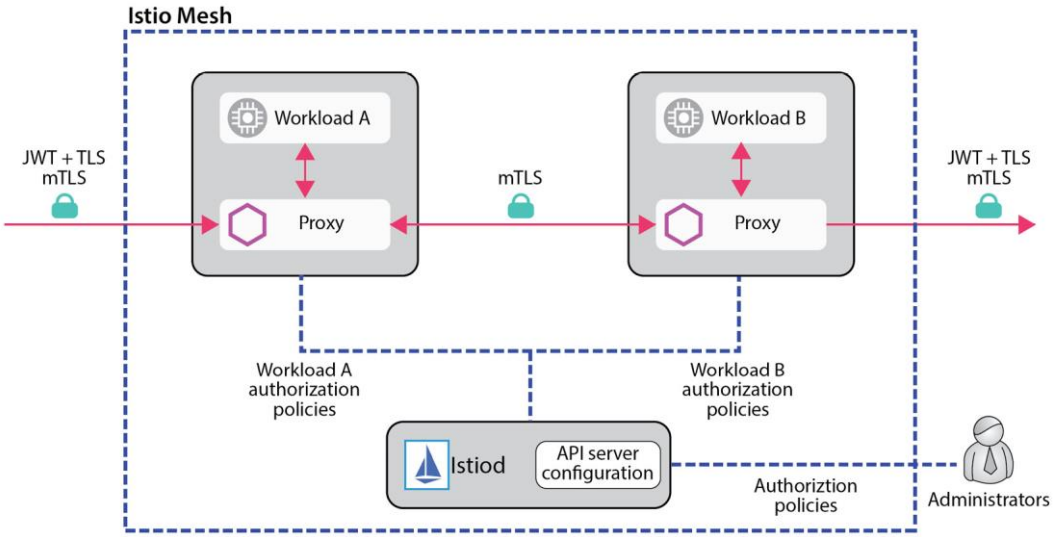


Figure 6.10 – Service-to-service authentication with Istio

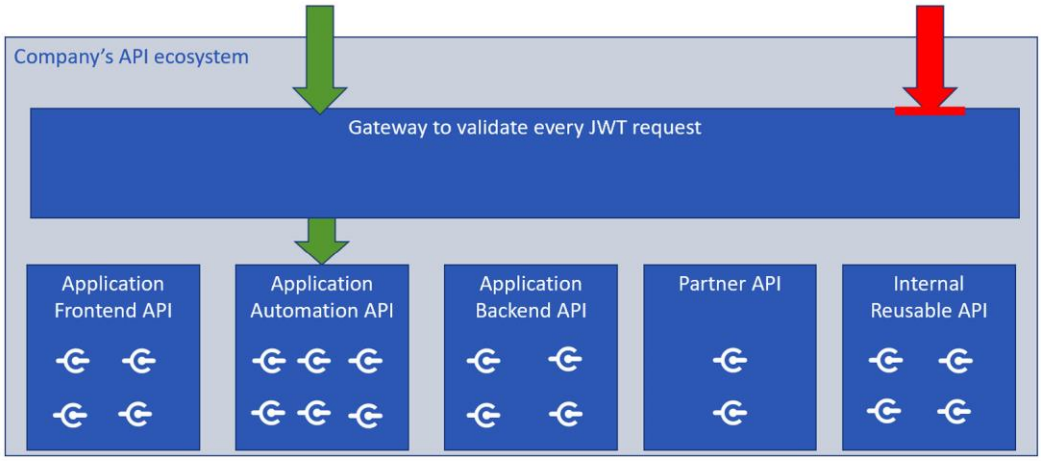


Figure 6.11 – Force check on every request

Chapter 7

Figures

Figure 7.1 – The Gartner Magic Quadrant for access management

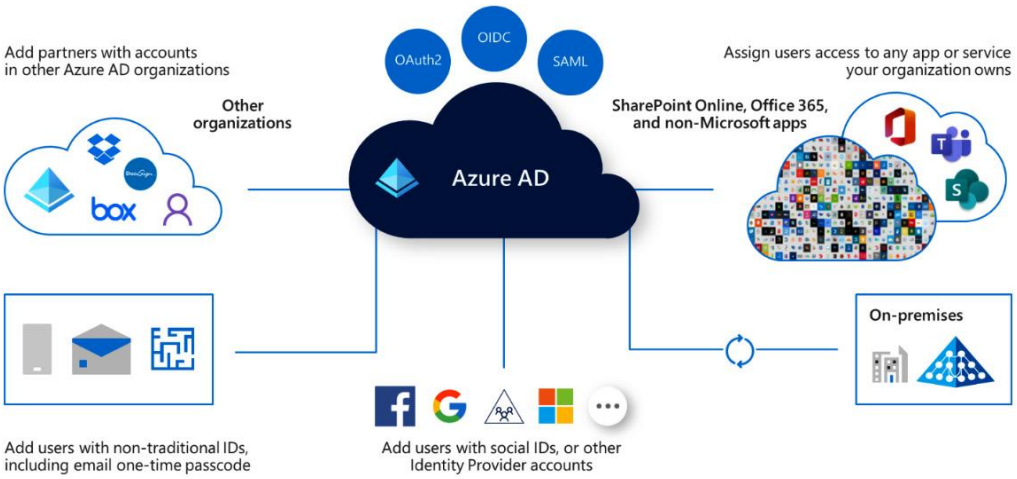


Figure 7.2 – AAD overview

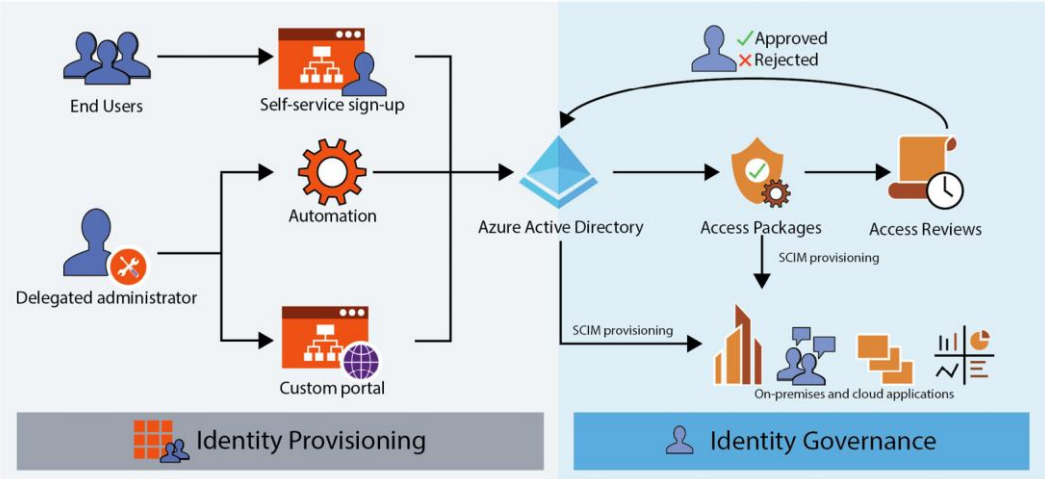


Figure 7.3 – Identity and access life cycle example

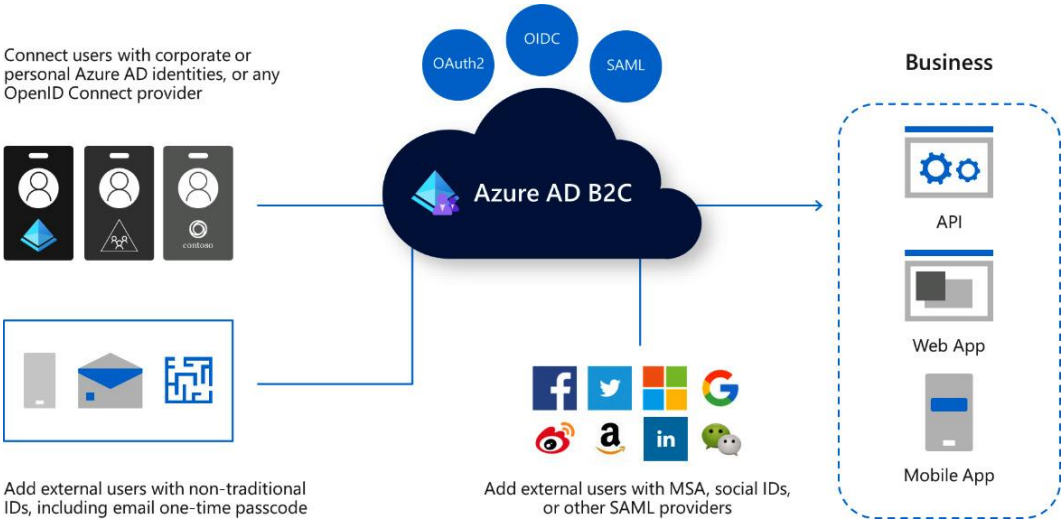


Figure 7.4 – AD B2C overview

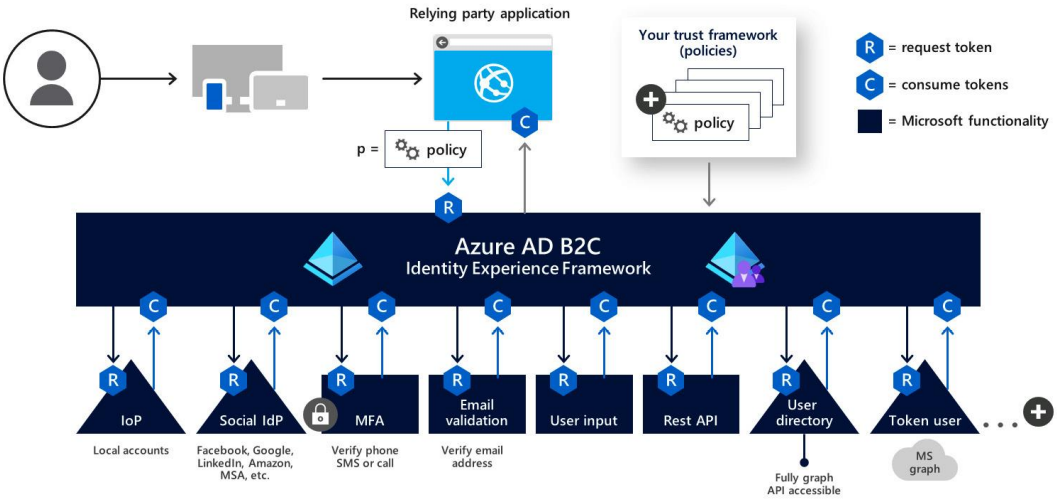


Figure 7.5 – AD B2C holistic overview

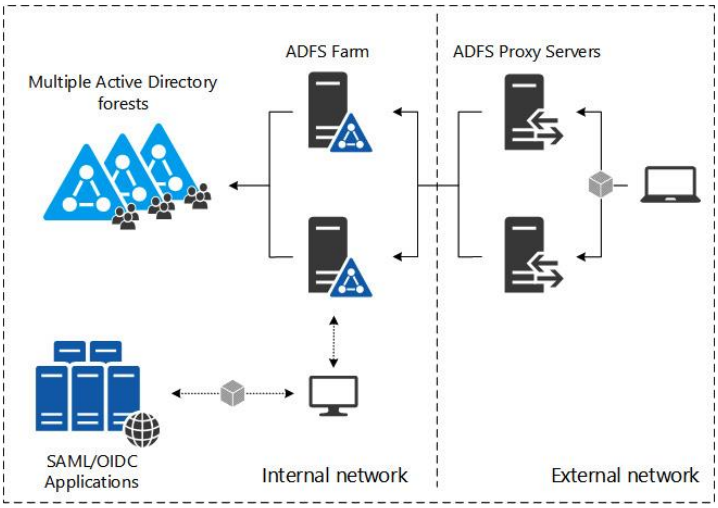


Figure 7.6 – ADFS architecture

Table

Decision factor	Description
Protocols	As mentioned multiple times in this book, each supported protocol enables a specific type of authentication workflow to be adopted by a specific company. This is usually one of the most important decision factors because protocols such as OIDC and OAuth 2.0 are a must. One caveat here is the way the OAuth specs are adopted and implemented by the IDP. This kind of evaluation may be pushed a level deeper to review not only the protocol implemented but also how close the IDP is to the RFC specs.
Federation	This is the ability of a specific IDP to be federated with another IDP using a protocol or out-of-the-box features such as the ability to invite external users from different IDPs as guests in the target IDP.
Multiple authentication factors	In the cloud era, with fewer network perimeters and more users, greater security depends on how authentication is protected. Having multiple factors of authentication is rapidly becoming a minimum requirement in many companies to enable users to be authenticated not only via a password but also via an SMS, a phone call, an email, certificates, authenticator applications, and other authentication systems.
Client-side libraries	When developers need to write an application that relies on a specific IDP for authentication, the support of the client-side libraries is an important factor. It's important to note here that many client-side libraries are protocol-based and not IDP-based. There are plenty of IDP-based client-side libraries that take advantage of specific features of IDPs. It's important that these libraries cover a wide range of coding languages.
Documentation	This is an important factor for all the technology choices, not just the IDP. Having good documentation means providing good support and references

	so that the developers can write code in a standard way and, most importantly, find answers to common problems. The documentation topic is not only for developers but, importantly, it also needs to be followed by the IDP administrator within the company to find answers and be supported in daily tasks (for example, guiding the registration of an application on an OAuth IDP).
Cost	Although it may not sound like a technical factor, the cost is still relevant because if the cost is low but the maintenance time required to support the IDP within the company is high, a low cost does not imply a better choice. The cost factor is usually a trade-off between the money requirement of the vendor and the overhead the company has to take into account to support the IDP operations.
Support	The availability of the IDP vendor to support the enterprise that chose to adopt a specific IDP is an important factor. This support is generally provided in various pricing tiers according to the SLA the enterprise requires.
Claims	The ability to affect claims before the token is forged by the IDP is typically an important technical aspect that needs to be considered. It is common for an enterprise to require a specific claim to be transformed or injected in order to be read by the final application, which needs to consume an access token or ID token.
API	APIs are usually required, especially by large enterprises, to automate different tasks (for example, application registration or renewing secrets). If APIs are not provided, the tasks need to be done manually, and this could be time-consuming, especially for big enterprises.
Customization	This feature is especially important for enterprises that want to adopt the IDP for their end customers. This functionality includes the ability to perform advanced actions, such as customizing the login screen for specific applications or creating and editing authentication workflows.
Hard limits	Each IDP has a limit in terms of how many users it can support or how many requests per second it can serve. This requires special attention from the enterprise architects, who need to be well aware of the requirements of the organization and the hard limits that can be tolerated for their application portfolio.
Auditing and reporting	Auditing and reporting is the ability to view reports of logins and logouts, enabling administrators to understand how many users access a specific application. One important example in this area is the ability to review suspicious activity: many IDPs raise an alert when, for example, a specific user logs in from the United States, and after a few minutes, the same user logs in from Europe in a timeframe not compatible with an actual relocation, which may indicate hacker activity.
Conditional access	Certain IDPs allow you to grant access to a specific resource not only based on the user identity but also based on what device the user is using, the

	region the user is connecting from, what their IP address is, or other custom criteria that can enhance the login security.
Provisioning	Users can be provisioned and de-provisioned in multiple ways. The bare minimum functionality is to have a portal that enables an administrator to provision new users. Then, there can be functionalities available via APIs or supported batch processes for onboarding or offboarding users from the IDP.
Self-service	This criterion encompasses all the actions that a user can do on their own without needing an administrator's help. Examples are self-service password resets (the user in this case is authorized by another factor), profile updates, or any self-management activities that can reduce the workload of the help desk.
Policies	An example of a policy could be a password policy such as configurable account expiration, the minimum number of characters, or the set of characters a password must contain. Policies are a broad concept and an IDP enables the administrator to establish advanced policies to manage it.
Extra features	This table thus far has covered the most typical technical factors of an IDP. Other features include temporary admin privileges, integrations, delegated administration, token signing, token encryption, and access restriction.

Table 7.1 – Technical decision factors

Links

NIST documents: https://pages.nist.gov/800-63-3/sp800-63b.html

Okta Workflows: https://www.okta.com/platform/workflows/

Chapter 8

Figures

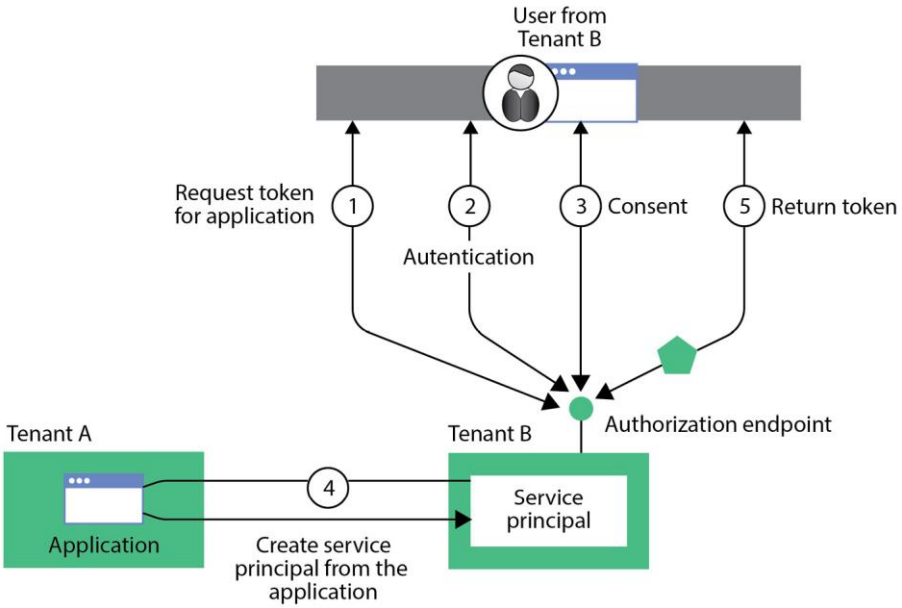


Figure 8.1 – Service principal creation flow in a multi-tenant AAD scenario

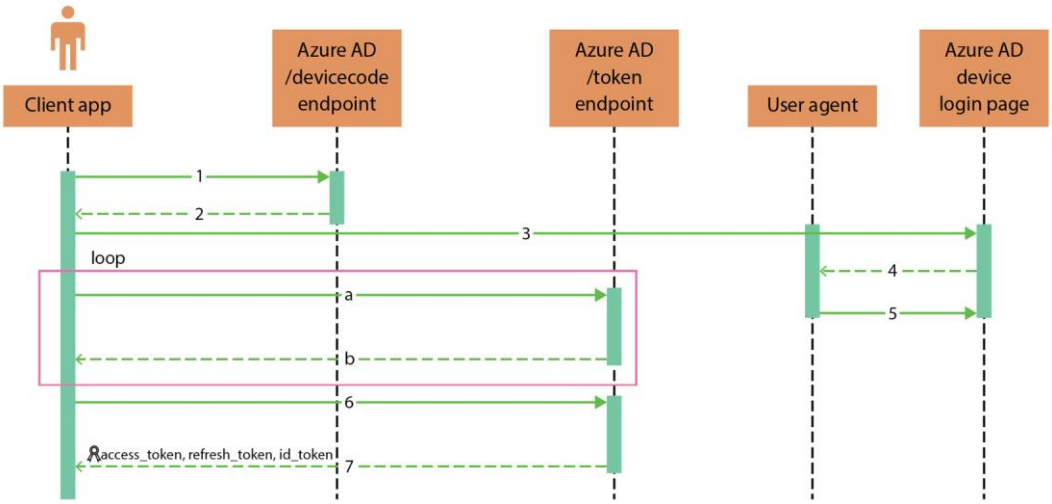


Figure 8.2 – Device authorization grant flow

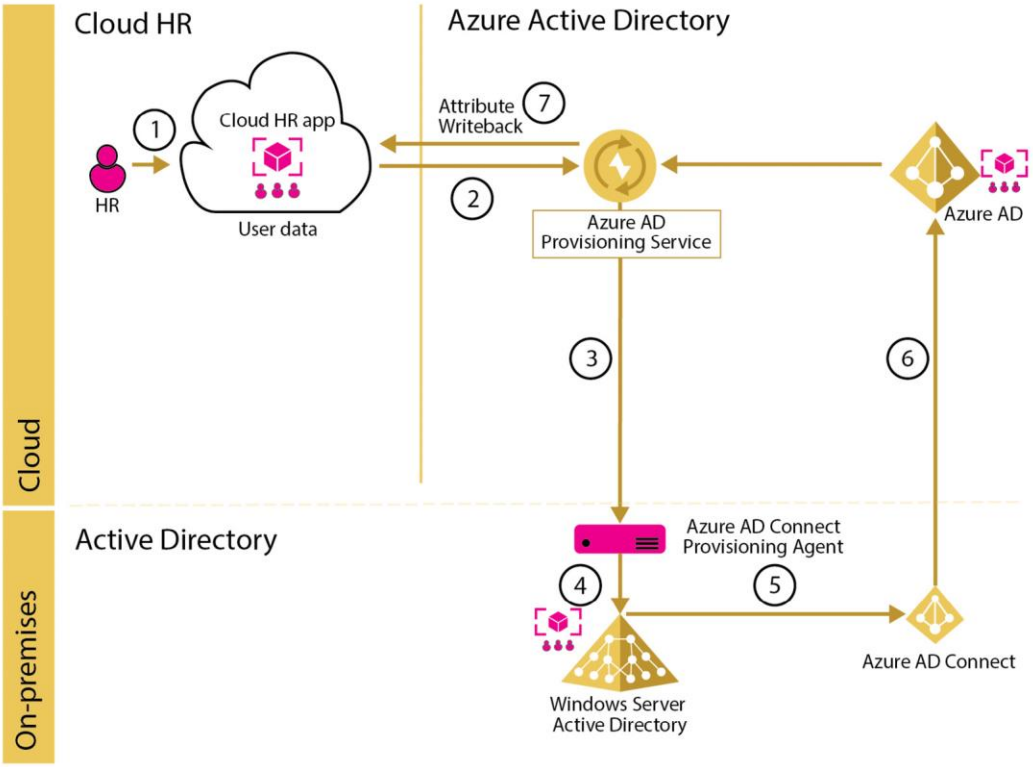


Figure 8.3 – Cloud HR app to on-premises synchronization flow

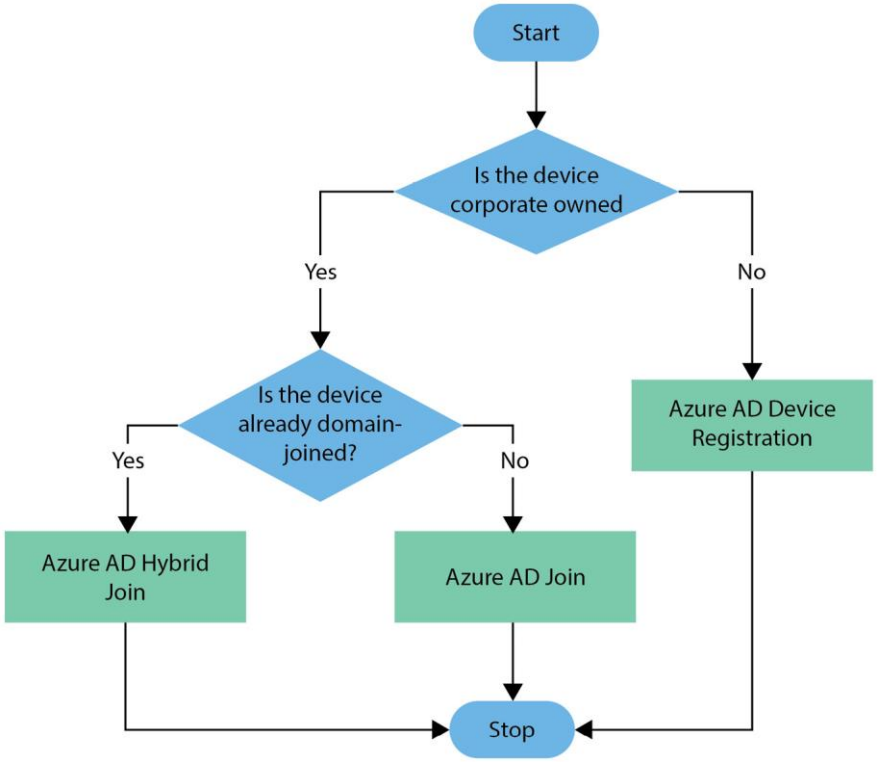


Figure 8.4 – AAD device onboarding decision flowchart

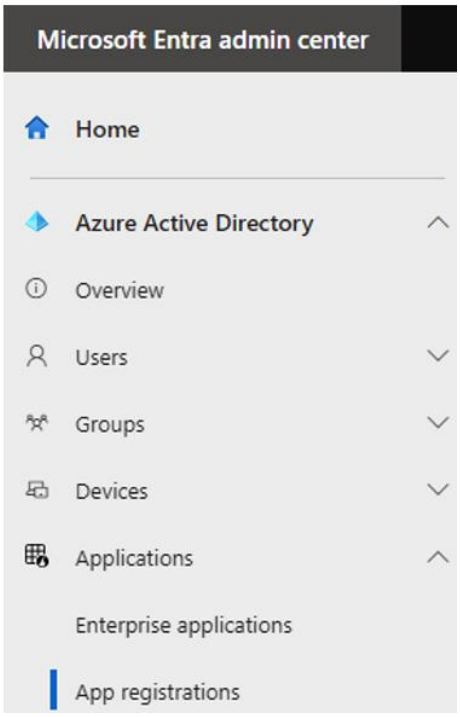


Figure 8.5 – Entra admin center

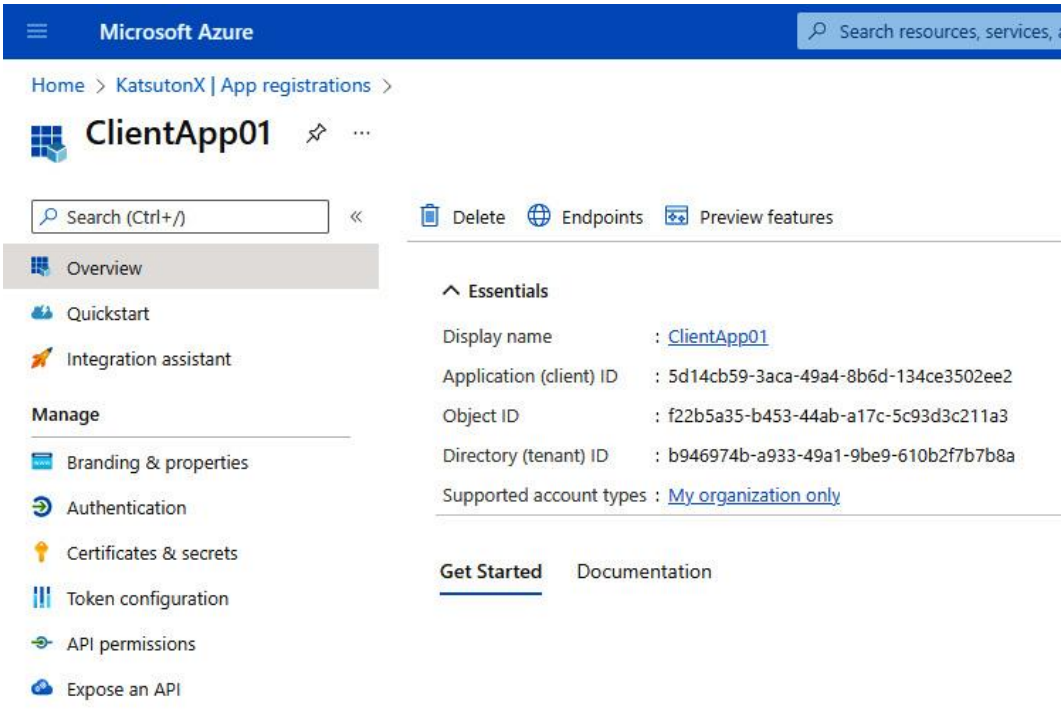


Figure 8.6 – An application definition overview

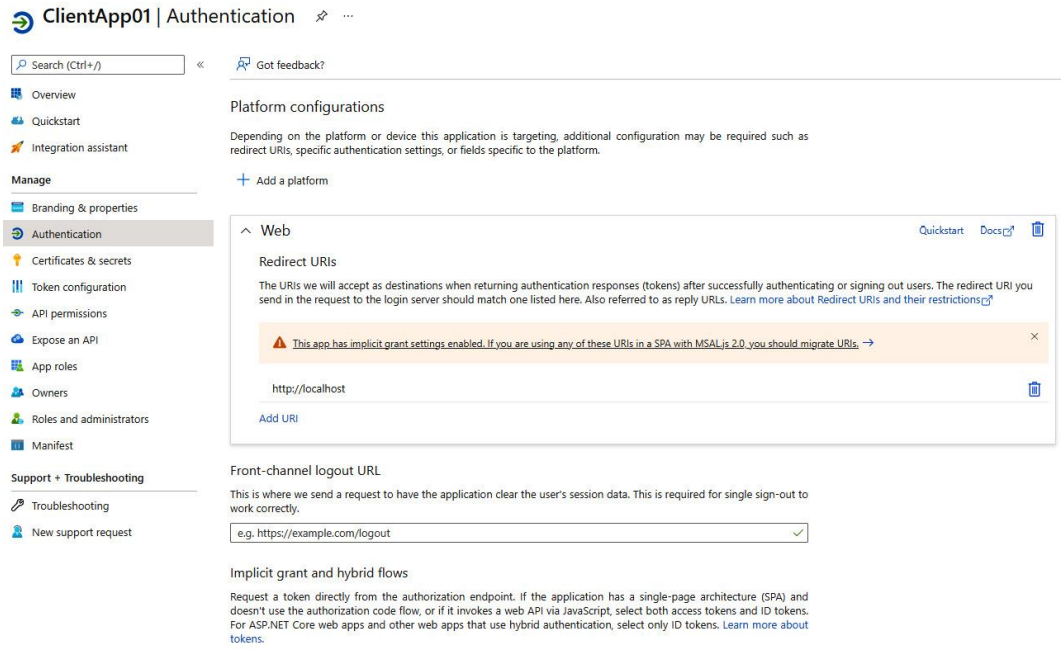


Figure 8.7 – Authentication menu for the ClientApp01 definition

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

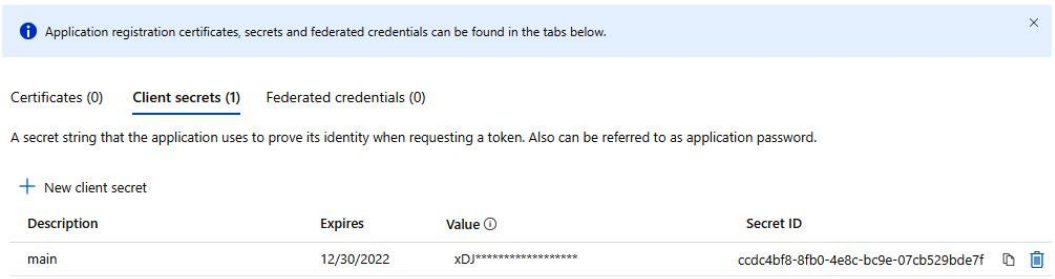
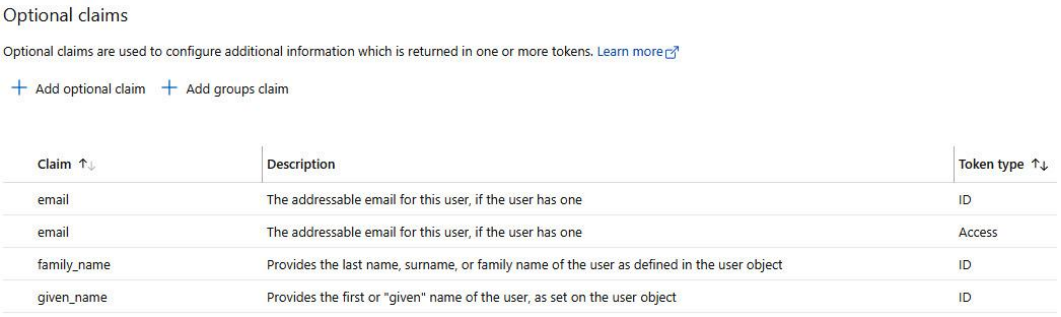


Figure 8.8 – Certificates and secrets menu for the ClientApp01 application



Claim	Description	Token type
email	The addressable email for this user, if the user has one	ID
email	The addressable email for this user, if the user has one	Access
family_name	Provides the last name, surname, or family name of the user as defined in the user object	ID
given_name	Provides the first or "given" name of the user, as set on the user object	ID

Figure 8.9 – Token configuration for the ClientApp01 application

Configured permissions

Applications are authorized to call APIs when they are granted permissions by users/admins as part of the consent process. The list of configured permissions should include all the permissions the application needs. Learn more about permissions and consent

+ Add a permission ✓ Grant admin consent for KatsutonX

API / Permissions name	Type	Description	Admin consent requ...	Status
Microsoft Graph (3)				
email	Delegated	View users' email address	No	Granted for KatsutonX
profile	Delegated	View users' basic profile	No	Granted for KatsutonX
User.Read	Delegated	Sign in and read user profile	No	Granted for KatsutonX
ServerApp01 (2)				
Read	Delegated	Read app data	No	Granted for KatsutonX
Write	Delegated	Write app data	Yes	Granted for KatsutonX

To view and manage permissions and user consent, try Enterprise applications.

Figure 8.10 – API permissions menu for the ClientApp01 definition

Application ID URI api://ceff2f83-b273-470c-8d05-23d04d877528

Scopes defined by this API

Define custom scopes to restrict access to data and functionality protected by the API. An application that requires access to parts of this API can request that a user or admin consent to one or more of these.

Adding a scope here creates only delegated permissions. If you are looking to create application-only scopes, use 'App roles' and define app roles assignable to application type. Go to App roles.

+ Add a scope

Scopes	Who can consent	Admin consent display ...	User consent display na...	State
api://ceff2f83-b273-470c-8d05-23d04d877528/Write	Admins only	Write app data		Enabled
api://ceff2f83-b273-470c-8d05-23d04d877528/Read	Admins and users	Read app data		Enabled

Authorized client applications

Authorizing a client application indicates that this API trusts the application and users should not be asked to consent when the client calls this API.

+ Add a client application

Client Id	Scopes
No client applications have been authorized	

Figure 8.11 – The Expose an API menu for the ServerApp01 definition

Endpoints

OAuth 2.0 authorization endpoint (v2)
https://login.microsoftonline.com/b946974b-a933-49a1-9be9-610b2f7b7b8a/oauth2/v2.0/authorize

OAuth 2.0 token endpoint (v2)
https://login.microsoftonline.com/b946974b-a933-49a1-9be9-610b2f7b7b8a/oauth2/v2.0/token

OAuth 2.0 authorization endpoint (v1)
https://login.microsoftonline.com/b946974b-a933-49a1-9be9-610b2f7b7b8a/oauth2/authorize

OAuth 2.0 token endpoint (v1)
https://login.microsoftonline.com/b946974b-a933-49a1-9be9-610b2f7b7b8a/oauth2/token

OpenID Connect metadata document
https://login.microsoftonline.com/b946974b-a933-49a1-9be9-610b2f7b7b8a/v2.0/.well-known/openid-configuration

Microsoft Graph API endpoint
https://graph.microsoft.com

Federation metadata document
https://login.microsoftonline.com/b946974b-a933-49a1-9be9-610b2f7b7b8a/federationmetadata/2007-06/federationmetadata.xml

WS-Federation sign-on endpoint
https://login.microsoftonline.com/b946974b-a933-49a1-9be9-610b2f7b7b8a/wsfed

SAML-P sign-on endpoint
https://login.microsoftonline.com/b946974b-a933-49a1-9be9-610b2f7b7b8a/saml2

SAML-P sign-out endpoint
https://login.microsoftonline.com/b946974b-a933-49a1-9be9-610b2f7b7b8a/saml2

Figure 8.12 – AAD protocol endpoints

Permissions

Applications can be granted permissions to your organization and its data by three methods: an admin consents to the application for all users, a user grants consent to the application, or an admin integrating an to the application. Learn more.

To request additional permissions for this application, use the application registration.

As an administrator you can grant consent on behalf of all users in this tenant, ensuring that end users will not be required to consent when using the application. Click the button below to grant admin consent.

Grant admin consent for KatsutonX

Admin consent | User consent

Search permissions

API Name	Claim value	Permission	Type
Microsoft Graph			
Microsoft Graph	email	View users' email address	Delegated
Microsoft Graph	profile	View users' basic profile	Delegated
Microsoft Graph	User.Read	Sign in and read user profile	Delegated
ServerApp01			
ServerApp01	Write	Write app data	Delegated
ServerApp01	Read	Read app data	Delegated

Figure 8.13 – ClientApp01 service principal's granted permissions (scopes)

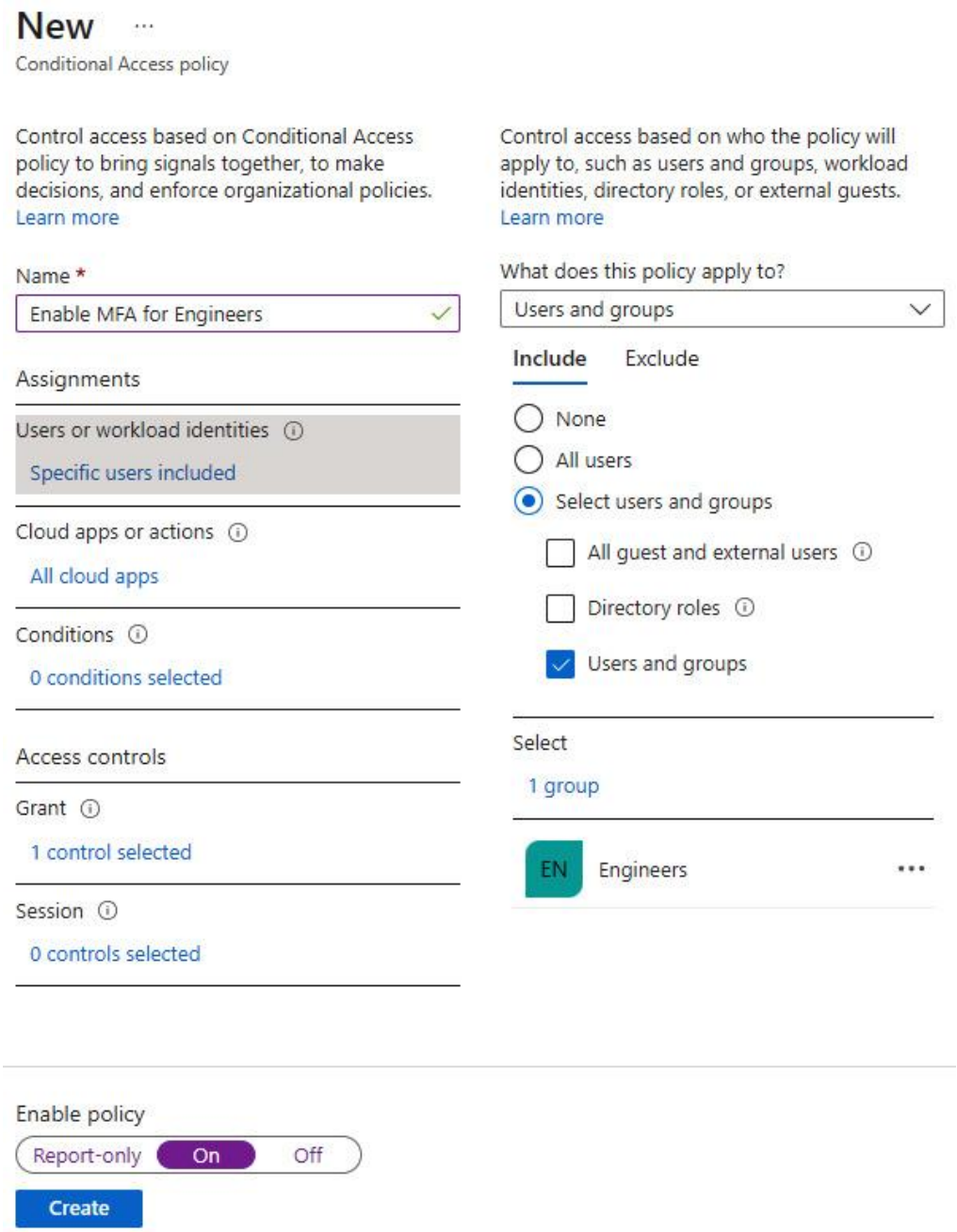


Figure 8.14 – Conditional Access policy creation

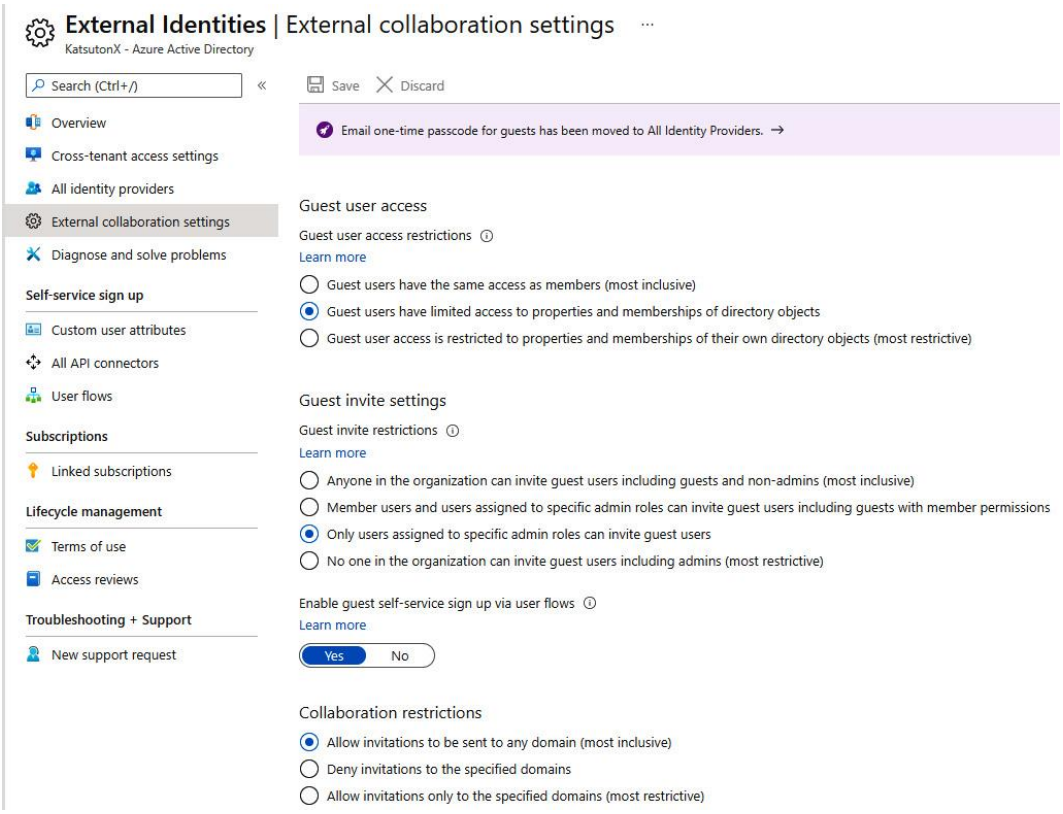


Figure 8.15 – AAD external collaboration settings

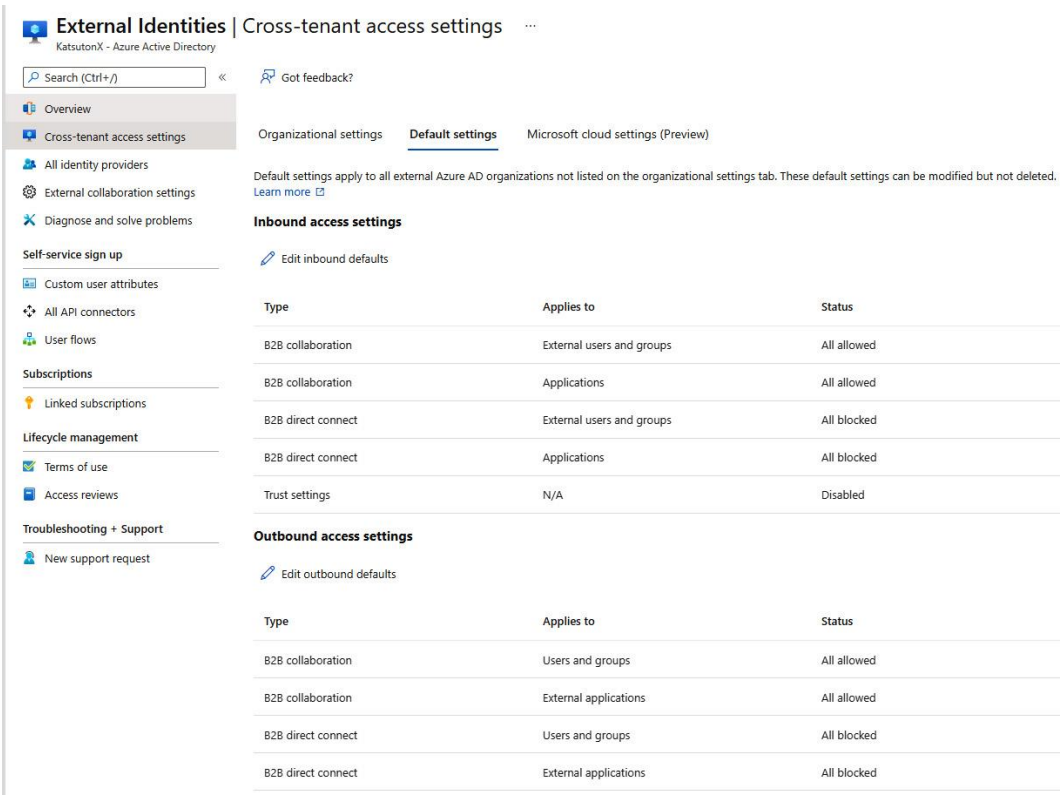


Figure 8.16 – AAD cross-tenant access settings

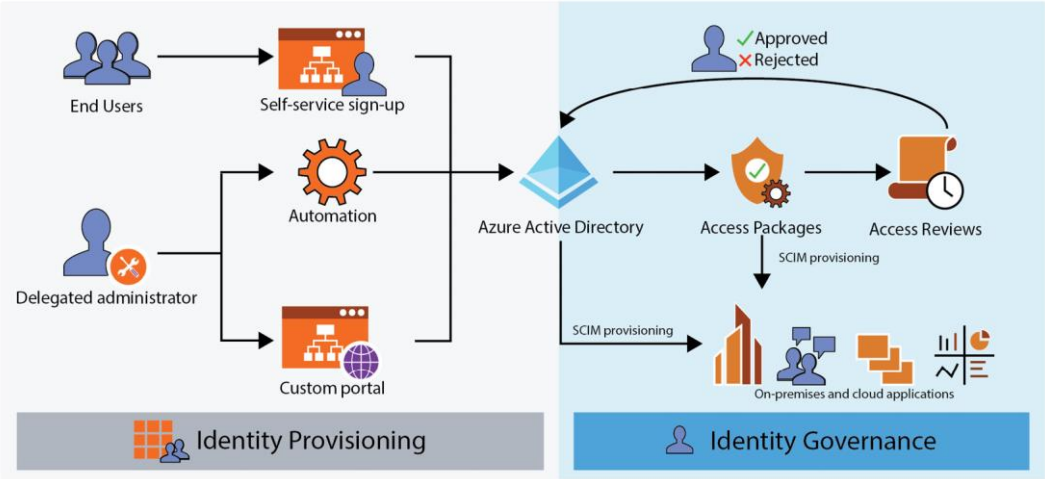


Figure 8.17 – An AAD user's life cycle

Table

	MFA	Federated	Pass Through	Seamless SSO	Password Hash Sync	Passwordless	TAP
Cloud user	Yes	N/A	N/A	N/A	N/A	Yes	Yes
Hybrid user	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 8.1 – AAD users' authentication types

Method	Description	HTTP verb
List users	Get a list of user objects	**GET**
Create user	Create a new user object	**POST**
Get user	Read the properties and relationships of the user object	**GET**
Update user	Update the user object	**PATCH**
Delete user	Delete the user object	**DELETE**

Table 8.2 – Microsoft Graph user object methods

Code

Code 8.1: Client application sends a request to the AAD /devicecode

```
POST
https://login.microsoftonline.com/{tenant_name}/oauth2/v2.0/de
vicecode

Content-Type: application/x-www-form-urlencoded

client_id=s6BhdRkqt3

&scope=openid%20resource_server_id
```

Code 8.2: Example of an AAD-issued token

By taking as a reference the OAuth 2.0 configuration of the **ClientApp01** and **ServerApp01** applications described in the previous paragraph when discussing what a registered application was, here is an example of an AAD-issued access token obtained through the client credentials flow:

- **Request**:

```
GET https://login.microsoftonline.com/b946974b-a933-49a1-9be9-
610b2f7b7b8a/oauth2/v2.0/token HTTP/1.1

Host: login.microsoftonline.com

Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_id=5d14cb59-3aca-49a4-
8b6d-
134ce3502ee2&client_secret=xDJ8Q~6QrdkVphZjTWkb9krIp5eQXGh8qEO
8rdyg&scope=api%3A%2F%2Fceff2f83-b273-470c-8d05-
23d04d877528%2F.default
```

- **Response body**:

```
{"token_type":"Bearer","expires_in":3599,"ext_expires_in":3599
,"access_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1Ni
IsIng1dCI6IjJaUXBKM1VwYmpBWVhZR2FYRUpsOGxWMFRPSSIsImtpZCI6
IjJaUXBKM1VwYmpBWVhZR2FYRUpsOGxWMFRPSSJ9.eyJhdWQiOiJhcGk6Ly
9jZWZmMmY4My1iMjczLTQ3MGMtOGQwNS0yM2QwNGQ4Nzc1MjgiLCJpc3Mi
OiJodHRwczovL3N0cy53aW5kb3dzLm5ldC9iOTQ2Otc0Yi1hOTMzLTQ5YT
EtOWJlOS02MTBiMmY3YjdiOGEvIiwiaWF0IjoxNjU2NjA1NjUwLCJuYmYi
OjE2NTY2MDU2NTAsImV4cCI6MTY1NjYwOTU1MCwiYWlvIjoiRTJaZ1lEQTl
JbnJqQnU4RUR3c1ZuLzhHNXJNTUFBPT0iLCJhcHBpZCI6IjVkMTRjYjU5L
TNhY2EtNDlhNC04YjZkLTEzNGNlMzUwMmVlMiIsImFwcGlkYWNyIjoiMSIs
ImlkcCI6Imh0dHBzOi8vc3RzLndpbmRvd3MubmV0L2I5NDY5NzRiLWE5MzM
tNDlhMS05YmU5LTYxMGIyZjdiN2I4YS8iLCJvaWQiOiI2NzI2M2NjYi1hOT
A5LTRkNWMtYjM2MC05ZmMyY2JkOGMzZTEiLCJyaCI6IjAuQVRzQVM1Zed1V
E9wb1VtYjZXRUxMM3Q3aW9Ndl84NXpzZ3hIalFVajBFMkhkU2c3QUFBLiIs
InN1YiI6IjY3MjYzY2NiLWE5MDktNGQ1Yy1iMzYwLTlmYzJjYmQ4YzNlMSI
sInRpZCI6ImI5NDY5NzRiLWE5MzMtNDlhMS05YmU5LTYxMGIyZjdiN2I4YS
IsInV0aSI6Ik5yTy16RHY4Q1Vxd1pRMVBMbEstQUEiLCJ2ZXIiOiIxLjAif
```

```
Q.Uv4SvZUSZye7LZvn93LrGtKVw3j-8TAZcVQgmlnX5_WWPx6LRPBrqh8sl
CC5y4hvHy1MV6rKbpepLSSo62U1-Ox6YxLyL_rxd3963Ue8P7voXAcvzQIt
veuPUDNe0Qhn7XPNuBvNIf-WiaSE-qSJE4n817qJ8cKXLNwNp-LqXCc38S9
NS6W0yRAE1qK5ZXOWgJbkBp0K5Td1NHA6hfBkaM7soDmvMuaADbGHRkGXxS
0QnxcLGaS-7s8VQjN82uG-XlnPEz6jMvvbLaomk1slCawYbTMqUgQqbB6mY
SEEhMVLZQ89hHbL3XpcDeFWgs45hBg6brlAIOzwV1JnPAaKRg"}
```

- **Decoded access token**:

```
{ "typ": "JWT", "alg": "RS256", "x5t":
"2ZQpJ3UpbjAYXYGaXEJl8lV0TOI", "kid":
"2ZQpJ3UpbjAYXYGaXEJl8lV0TOI" }.{ "aud": "api://ceff2f83-b273-
470c-8d05-23d04d877528", "iss":
"https://sts.windows.net/b946974b-a933-49a1-9be9-
610b2f7b7b8a/", "iat": 1656605526, "nbf": 1656605526, "exp":
1656609426, "aio": "E2ZgYDizmjtQvMnv0HdZwXUfBOYeAQA=",
"appid": "5d14cb59-3aca-49a4-8b6d-134ce3502ee2", "appidacr":
"1", "idp": "https://sts.windows.net/b946974b-a933-49a1-9be9-
610b2f7b7b8a/", "oid": "67263ccb-a909-4d5c-b360-9fc2cbd8c3e1",
"rh":
"0.AtsAS5dGuTOpoUmb6WELL3t7ioMv_85zsgxHjQUj0E2HdSg7AAA.",
"sub": "67263ccb-a909-4d5c-b360-9fc2cbd8c3e1", "tid":
"b946974b-a933-49a1-9be9-610b2f7b7b8a", "uti":
"0Y6Fozzet0aCs4veEArDAA", "ver": "1.0" }.[Signature]
```

In the *Registering and configuring applications* section, we explained the application management model that is implemented in AAD and its close relationship with the OAuth, OIDC, and SAML protocols. In the next section, we will go through a set of additional AAD features that integrate with its authentication capabilities through more enterprise-oriented features that enable collaboration scenarios with external organizations and increase the overall security posture of the organization.

Code 8.3: retrieve the list of properties associated with a user by leveraging the get user method:

- **Request**:

```
GET https://graph.microsoft.com/v1.0/users/me
```

- **Response**:

```
{
    "@odata.context":
"https://graph.microsoft.com/v1.0/$metadata#users/$entity",
    "businessPhones": [
        "4250000000"
    ],
    "displayName": "Fabrizio Barcaroli",
    "givenName": "Fabrizio",
    "jobTitle": null,
```

```
    "mail": "kadmin@katsutonx.onmicrosoft.com",
    "mobilePhone": null,
    "officeLocation": null,
    "preferredLanguage": null,
    "surname": "Barcaroli",
    "userPrincipalName": "kadmin@katsutonx.onmicrosoft.com",
    "id": "57f25ecb-d0ff-4707-8211-b1ab756696c0"
}
```

Links

What is a device identity? https://docs.microsoft.com/en-us/azure/active-directory/devices/overview.

AAD admin center (https://aad.portal.azure.com)

Microsoft Entra admin center (https://entra.microsoft.com)

Chapter 9

Figures

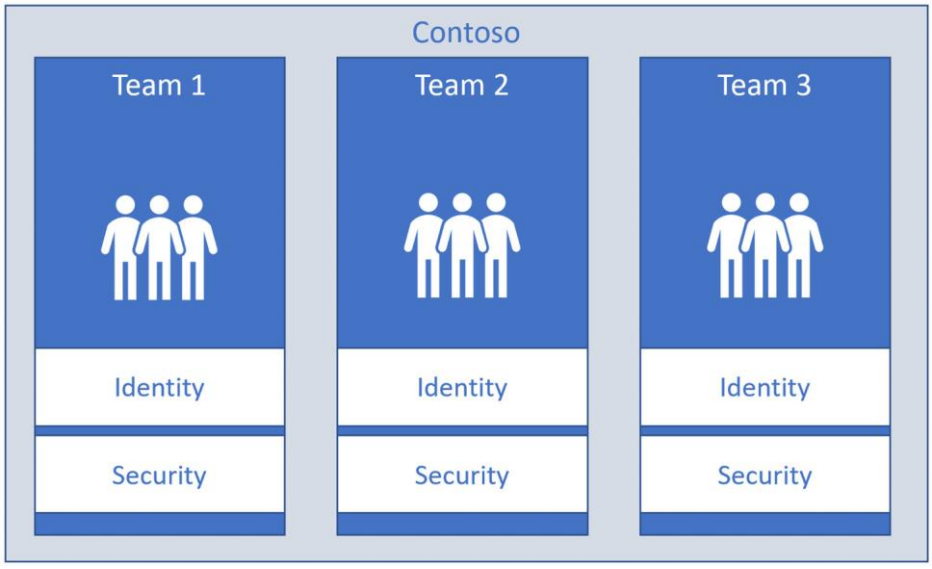


Figure 9.1 – Real-world example of independent teams, independent strategy

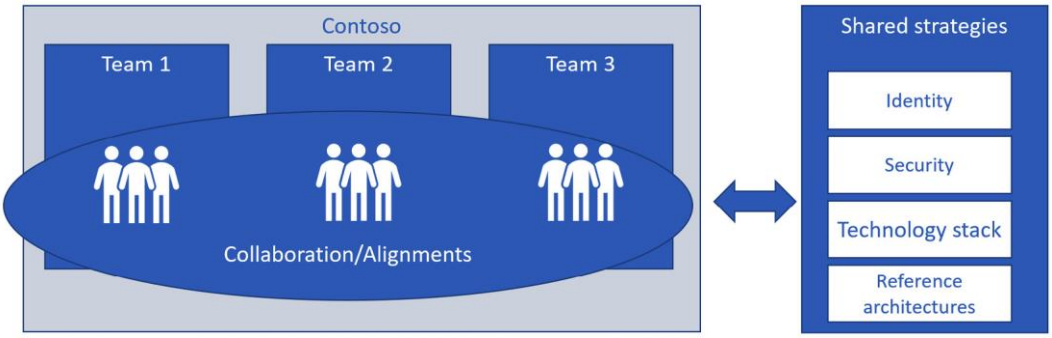


Figure 9.2 – Real-world example of independent teams, shared strategy

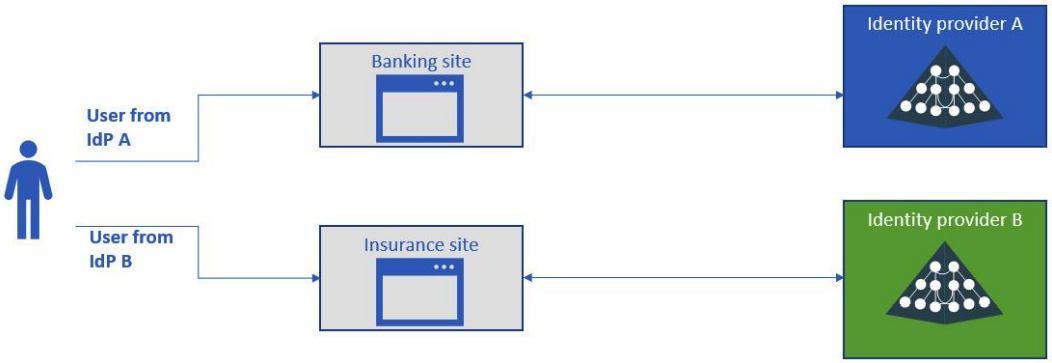


Figure 9.3 – Multiple customer-facing IdPs

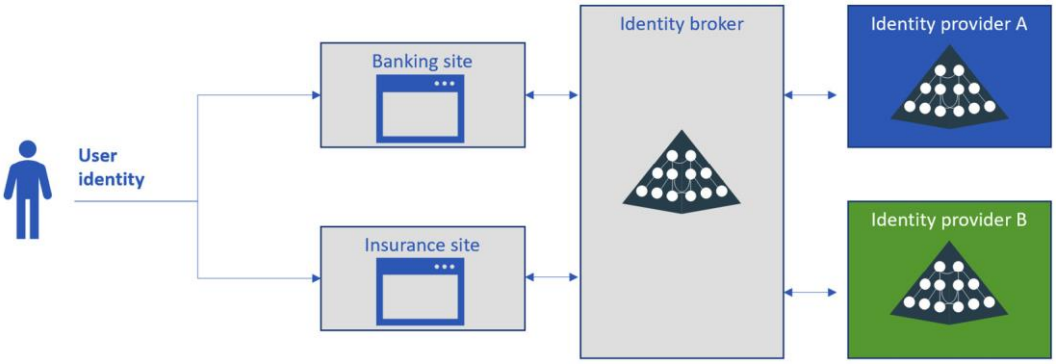


Figure 9.4 – Enhancing the UX with an identity broker

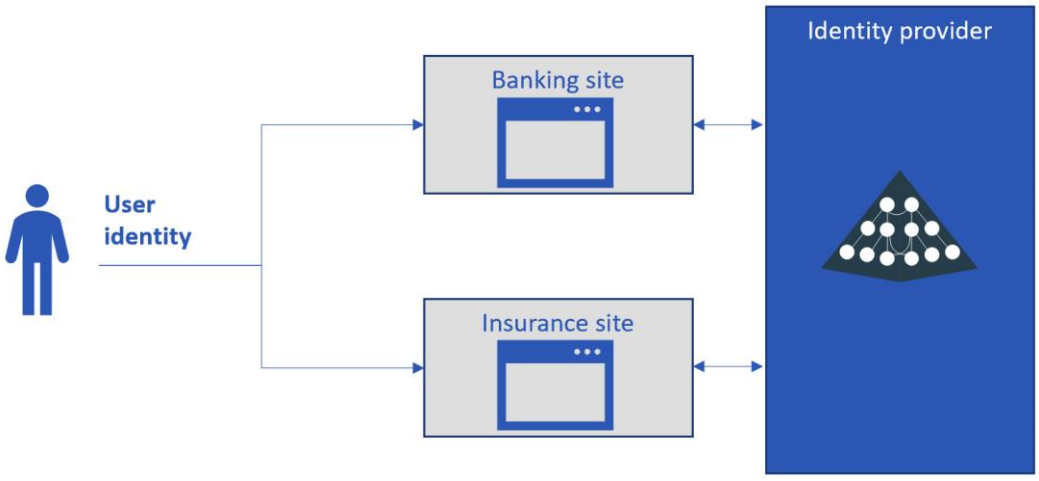


Figure 9.5 – A common IdP for consumers

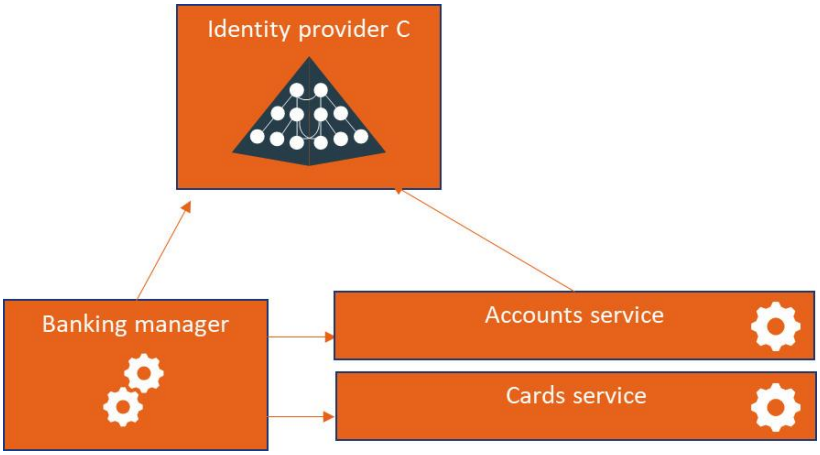


Figure 9.6 – Backend authentication sample on a modern application

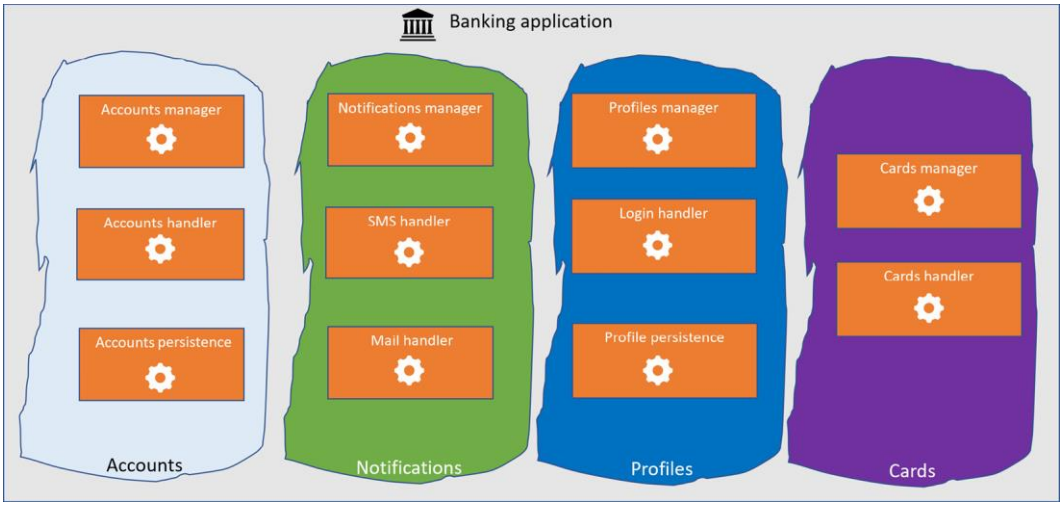


Figure 9.7 – Microservices application sample

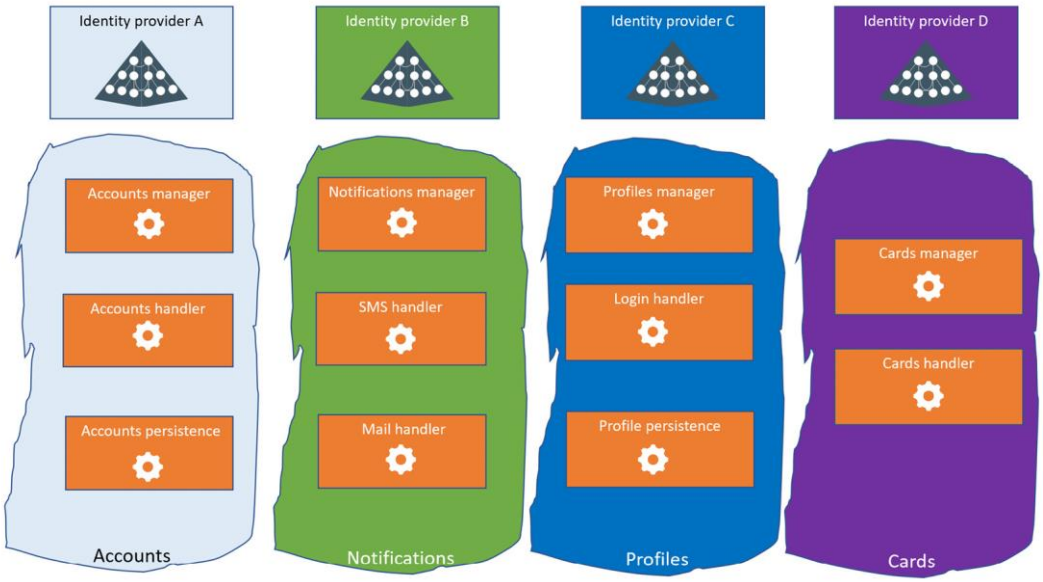


Figure 9.8 – Anti-pattern – multiple IdPs in different domain contexts

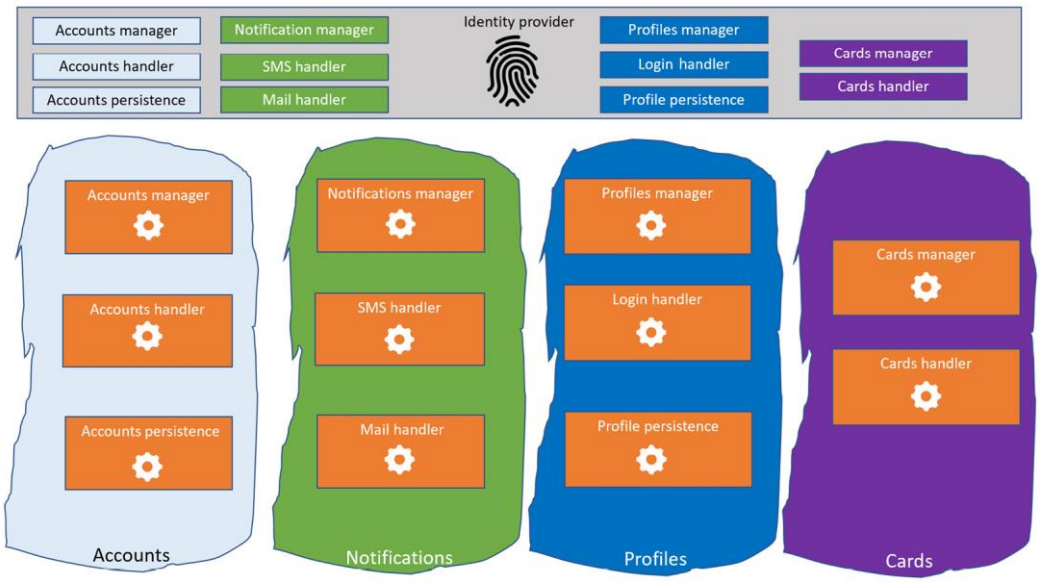


Figure 9.9 – Microservices application – single IdP

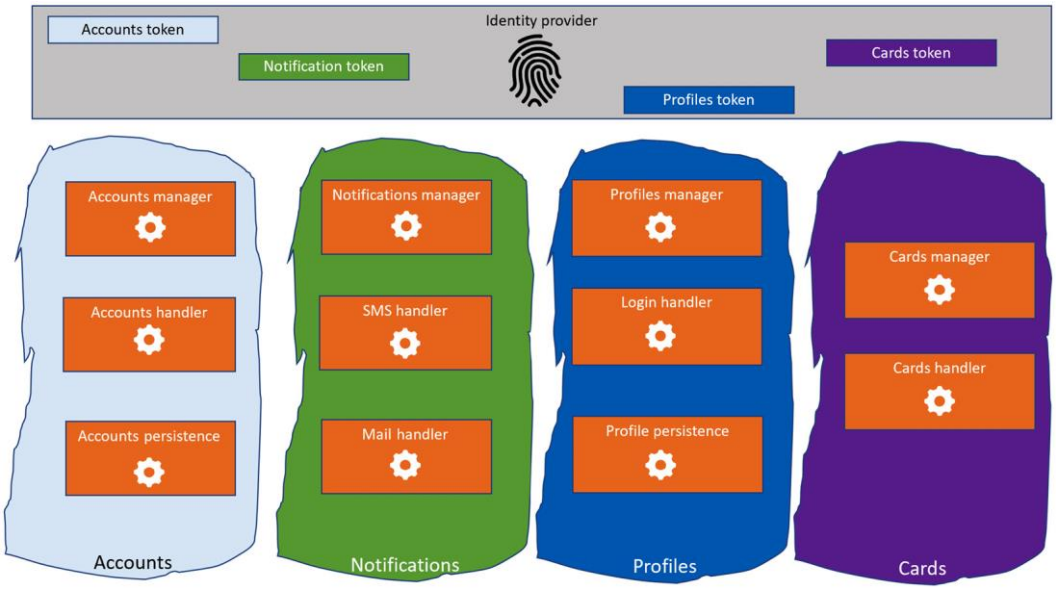


Figure 9.10 – Domain-based registration

Figure 9.11 – Application-based registration

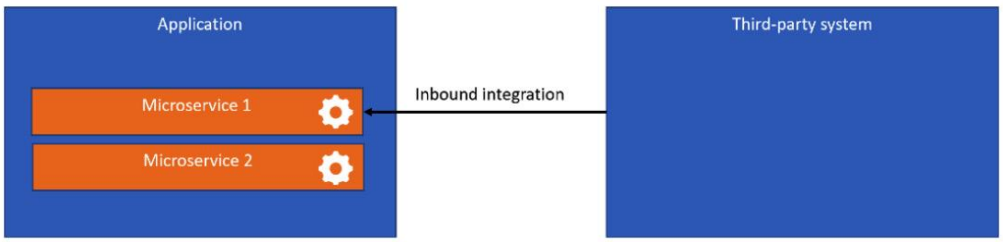


Figure 9.12 – Inbound integration

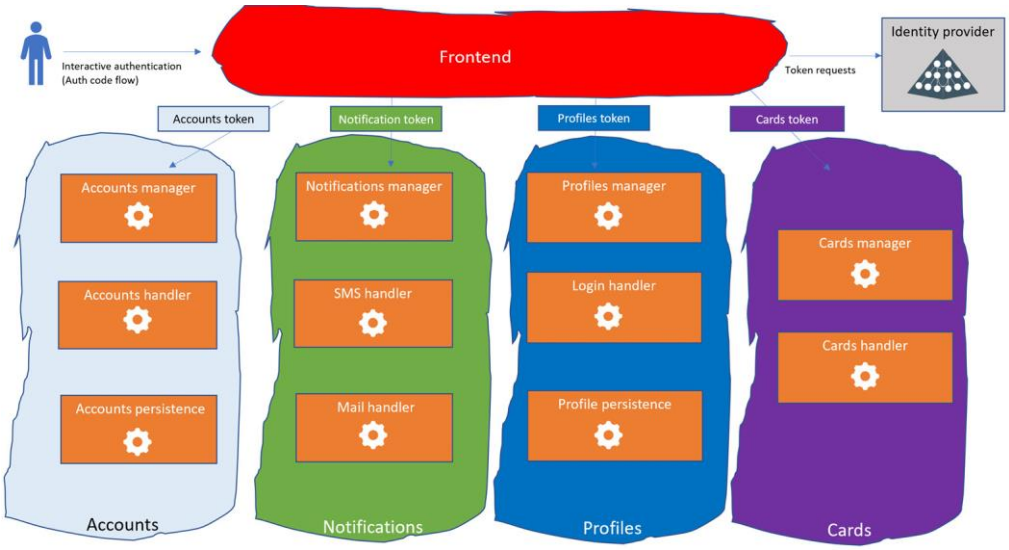


Figure 9.13 – Domain-based with a frontend component

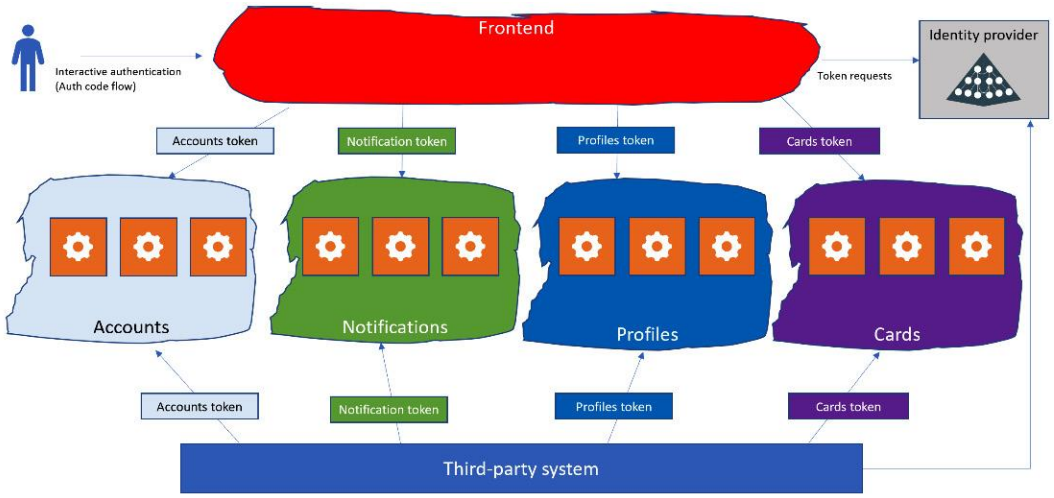


Figure 9.14 – Inbound integration anti-pattern

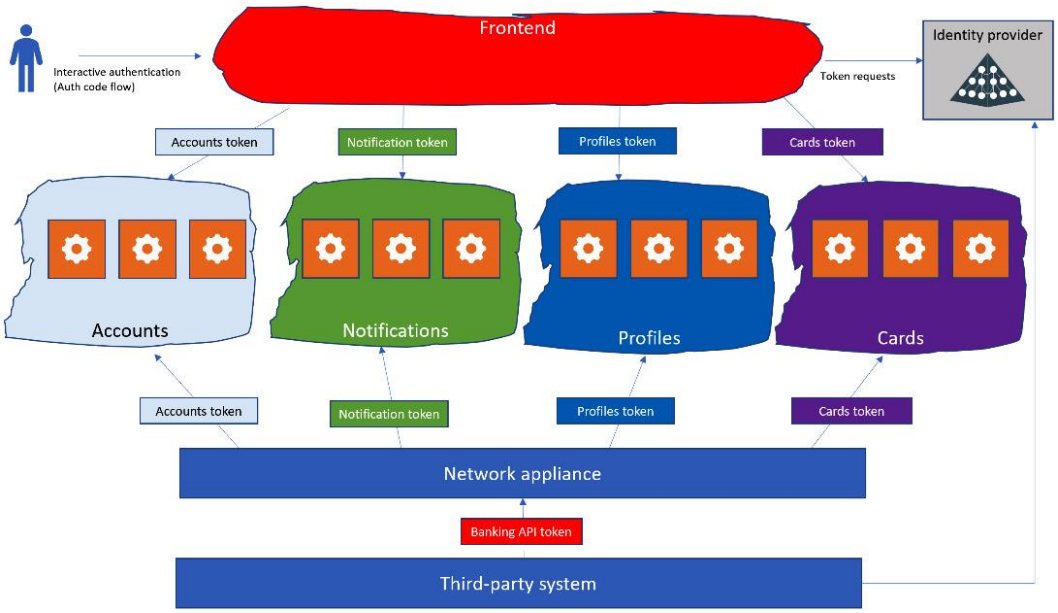


Figure 9.15 – Inbound integration pattern

Code

Code 9.1: The request for each token would be similar to the following snippet of an HTTP request:

```
POST /{tenant}/oauth2/v2.0/token HTTP/1.1
Host: hostname
Content-Type: application/x-www-form-urlencoded
client_id=1234567-8901-2345-6789-0123456678
&scope=AccountService
&client_secret=sampleCredentials
&grant_type=client_credentials
```

Links

Micro Frontends: https://martinfowler.com/articles/micro-frontends.html