

Zero Trust @ COA

Moving from Citrix to Azure.
The GUI for .NET frontend application IBIS at the COA organization had to comply to zero trust architecture.

Introduction



Richard Kelters

- Flusso employee since 2008
- 25+ years experience as a Progress OpenEdge developer
- PUG Netherlands board member from 2002 till 2018

Flusso

Flusso helpt organisaties om complexe IT om te zetten in overzicht, snelheid en grip.

- Met sterke expertise in onder andere:
 - bedrijfskritische systemen
 - moderne frontend en gebruikerservaring
 - low-code versnelling
 - cloud-native architecturen
- We helpen organisaties hun IT-landschap beheersbaar en toekomstklaar te maken
 - bestaande systemen moderniseren zonder verstoring
 - nieuwe applicaties slim en schaalbaar opbouwen
 - systemen laten samenwerken in complexe landschappen
 - technologie, processen en teams met elkaar verbinden

Hoe wij werken

- We leveren consistente kwaliteit door slimme inzet van technologie en tooling
 - AI-ondersteunde development voor hogere snelheid en kwaliteit
 - schaalbare cloudomgevingen op basis van Kubernetes
 - veilige en centrale identity oplossingen met Keycloak
 - versnelling waar het kan met low-code (OutSystems)
- Samenwerking bij Flusso is praktisch, transparant en gericht op resultaat
 - korte lijnen en duidelijke communicatie
 - gezamenlijke verantwoordelijkheid voor resultaat
 - oplossingen die werken in de praktijk, niet alleen op papier
 - duidelijke keuzes zonder onnodige complexiteit

Benieuwd hoe wij dit in de praktijk doen? Volg onze inzichten op LinkedIn.

Agenda

- COA organization
- IBIS application
- Architecture
- APSV challenge
- Azure Entra
- PASOE
- Proxy
- Demo
- Questions



COA *Centraal Orgaan opvang asielzoekers*

COA

Central Organization for Assylum seekers

- Governmental organization
- Justice department's responsibility
- Headquarters in The Hague
- 300 locations
- 75,000 residents
- 10,000 employees



IBIS

Integraal **B**woner **I**nformatie **S**ysteem
or
Integrated Resident Information System

- 2000 combining different applications into one ADM2 clientserver app
- 2008 start modernizing to GUI for .NET in version 10.2b using AppServer
- 2018 fully migrated
- 2021 migrated to pasoe version 11.7
- 2024 OpenEdge 12.2 WebClient
- 2025 OpenEdge 12.8

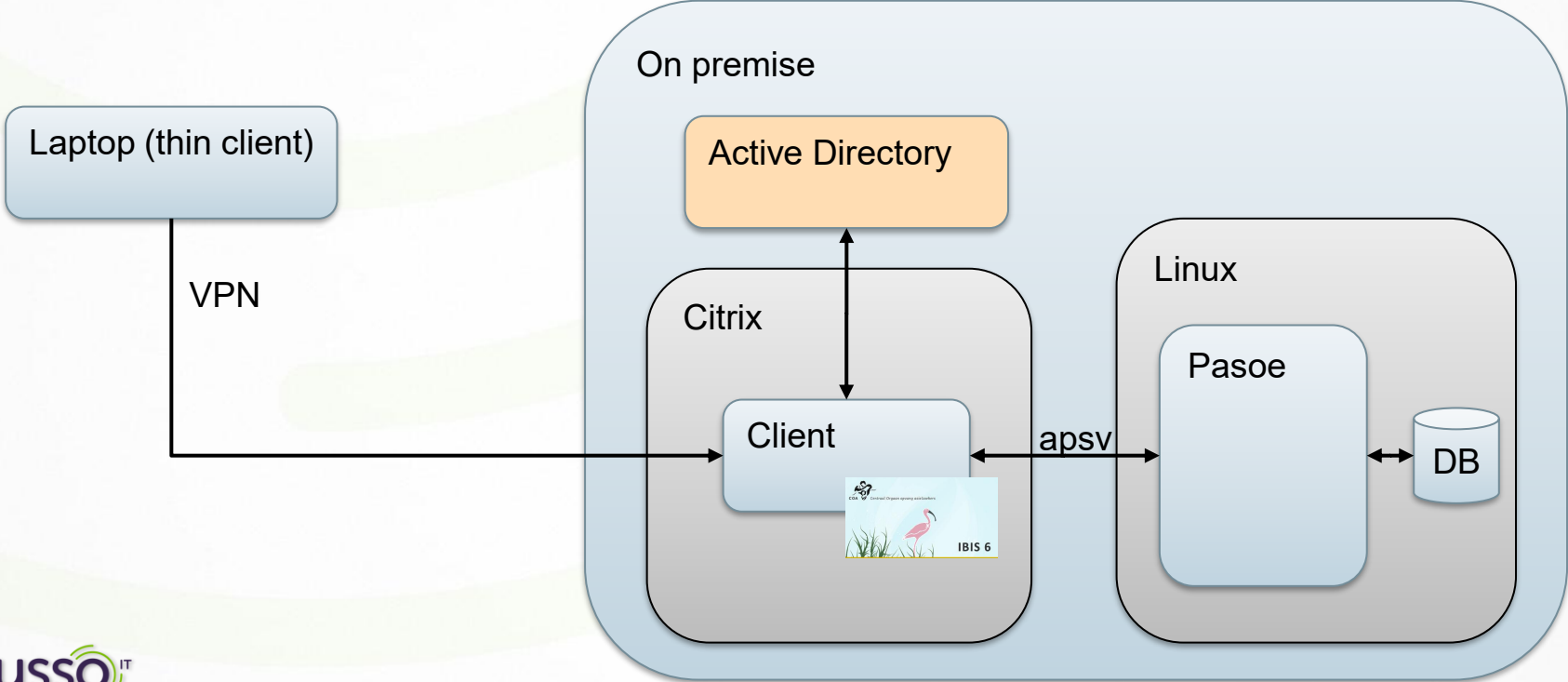


IBIS

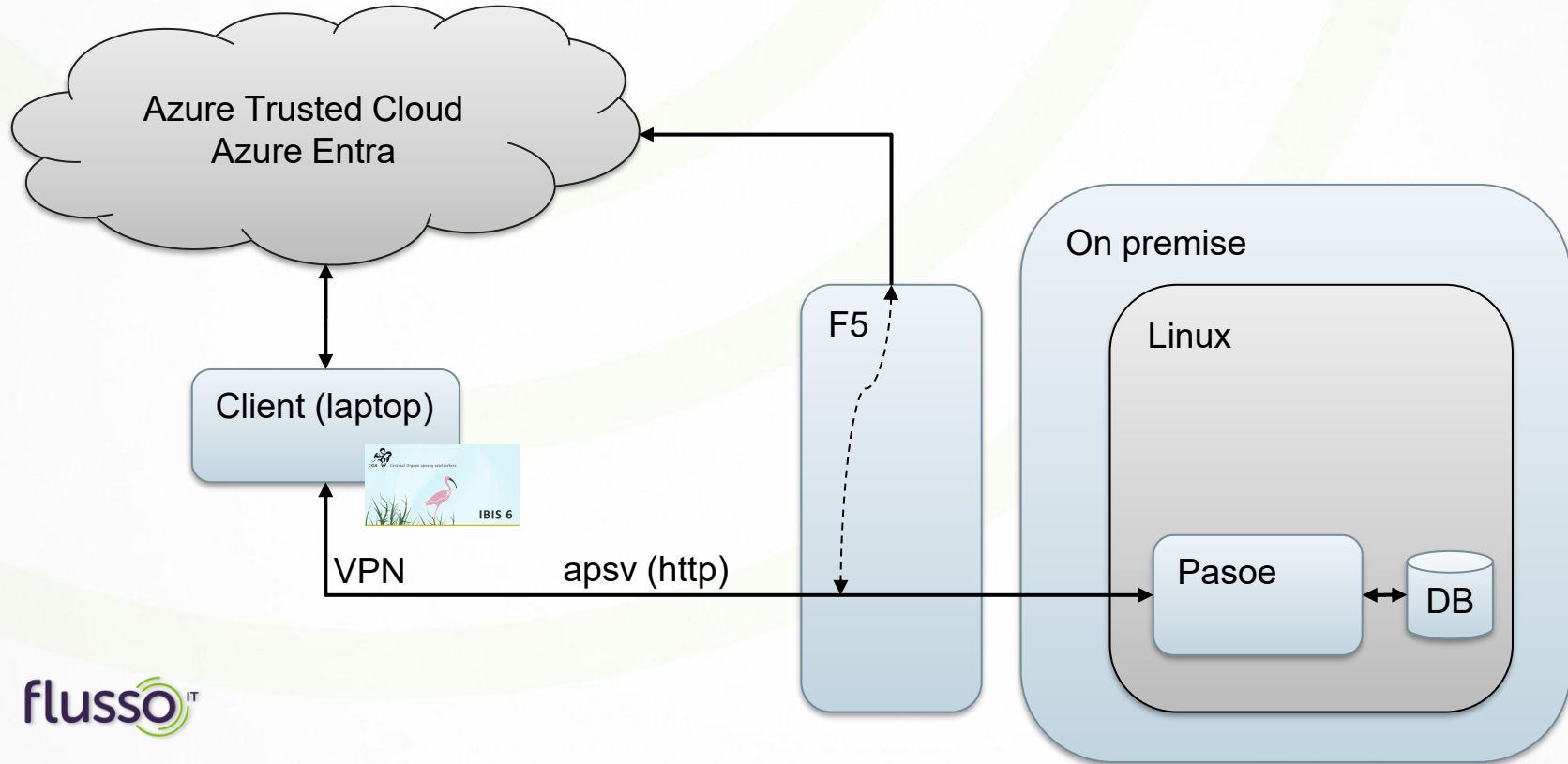
Integraal **B**woner **I**nformatie **S**ysteem

- 8000 active users
- 8 developers
- 4 databases
- IBIS database 150 GB
- 2 pasoe instances
- 3400 classes
- 1780 procedural programs
 - 950 database triggers
 - 90 Smart Data Objects
 - 200 service interfaces

Current architecture



Zero Trust architecture



APSV protocol

Proprietary protocol from Progress for OpenEdge

- Classic Appserver (V8 – V11)
- Progress Appserver for OpenEdge (V11-?)
 - tomcat
 - uses http
 - network packages in the body

```
create server appserver.
```

```
appserver:connect("-AppService IBIS -S 20211 -H codex ").
```

```
appserver:connect("-URL https://codex:20211").
```

APSV challenge

COA requested us to add an authorization header in all HTTP requests to comply to the zero trust architecture.

But

We don't have access to http requests or responses as an OpenEdge developer

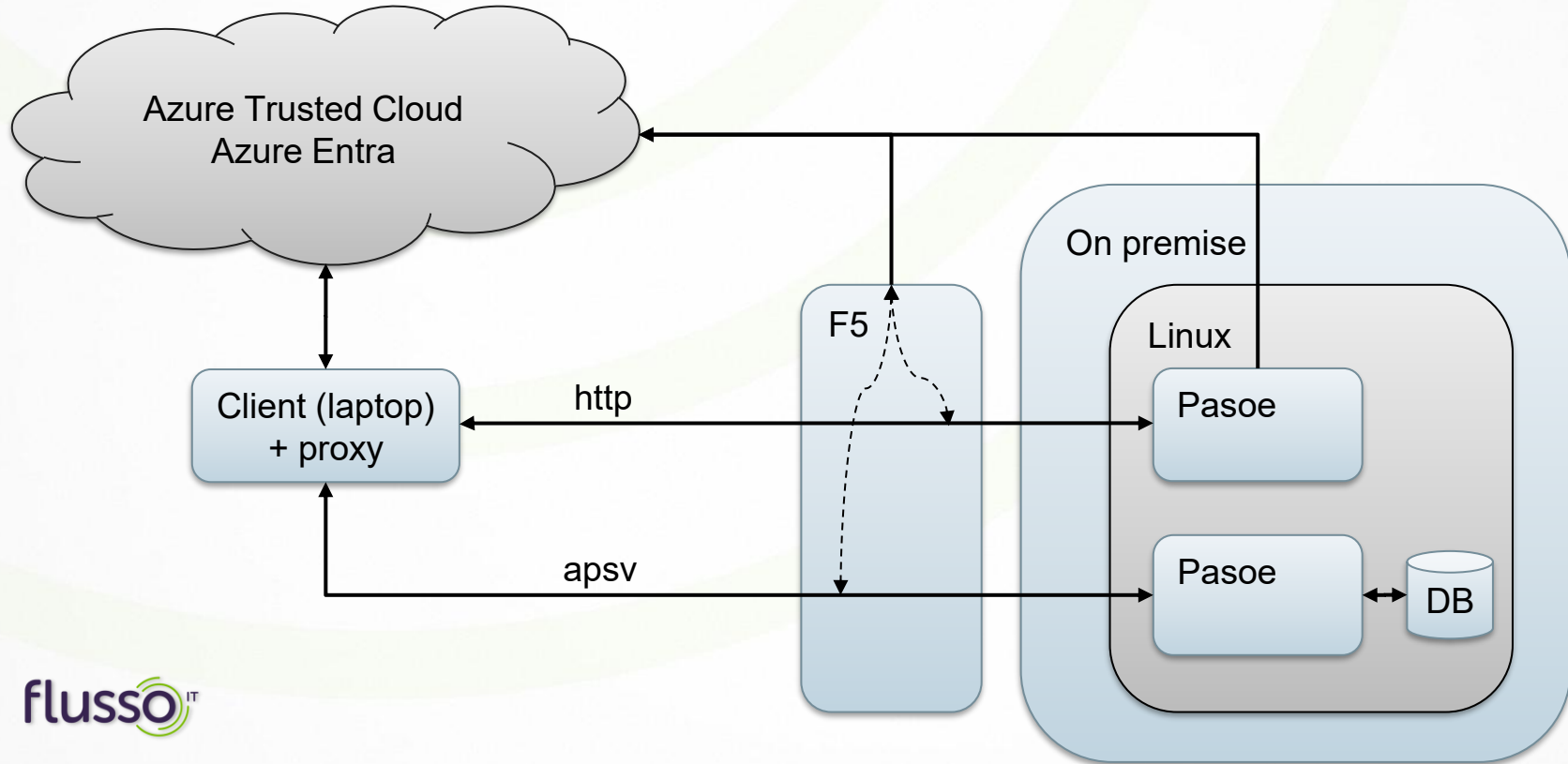
COA options for APSV

- Make an exception and allow unauthorized http requests for IBIS
- Wait for Progress Software to support http authorization for APSV
- Rewrite all APSV calls to http requests
- Make solution to add Authorization header to HTTP

Decision

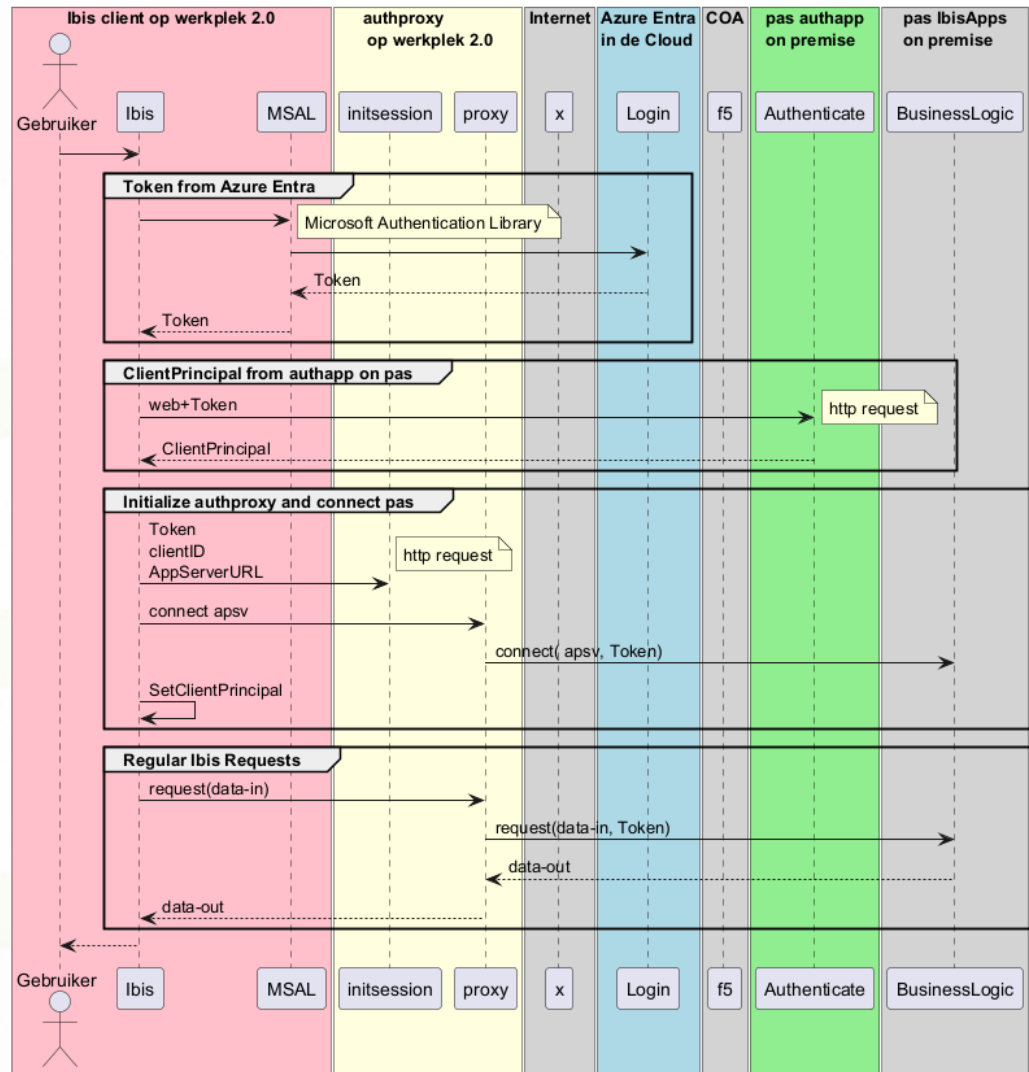
- Build a client proxy to inject Authorization header
- With language Go
 - Strong in networking
 - Stand alone executable

Zero Trust architecture



New login flow

- Azure Entra SSO
- Receive JWT
- Fetch ClientPrincipal
- Setup proxy
- Connect proxy
- SetClientPrincipal
- Ready

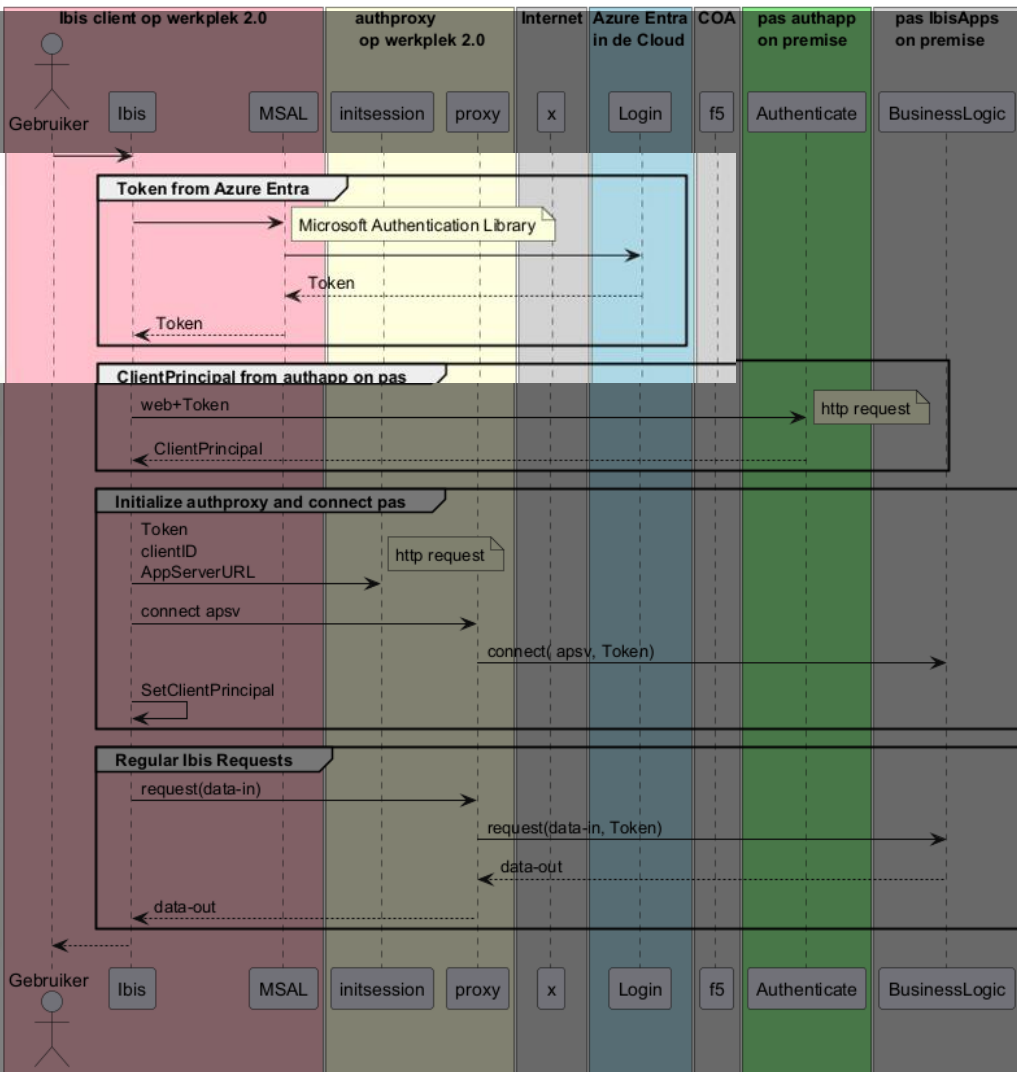


Challenges

- Acquire a JWT from Azure Entra
- Preserve current IBIS authentication with client-principal
- Setup pasoe with jwt validation
return IBIS specific client-principal
- Build a proxy
to add Authorization

New login flow

- Azure Entra SSO
- Receive JWT
- Fetch ClientPrincipal
- Setup proxy
- Connect proxy
- SetClientPrincipal
- Ready



Azure Entra AppRegistration

- Single Sign On
- Desktop application
- Provide callback URL
<https://localhost:12345>
- Supply userID from AD for backward compatibility
`onpremissesamaccountname`

You get:

- TenantId
- ClientId

MSAL

Microsoft Authentication Library

<https://www.nuget.org/packages/Microsoft.Identity.Client/>

Microsoft.Identity.Client 4.79.1

Prefix Reserved

.NET 8.0 .NET Standard 2.0 .NET Framework 4.6.2

.NET CLI PMC PackageReference CPM Paket CLI Script & Interactive File-based Apps Cake

```
> #r "nuget: Microsoft.Identity.Client, 4.79.1"
```

Copy

#r directive can be used in F# Interactive and Polyglot Notebooks. Copy this into the interactive tool or source code of the script to reference the package.

README Frameworks **Dependencies** Used By Versions Release Notes

Microsoft Authentication Library (MSAL) for .NET

The MSAL library for .NET is part of the [Microsoft identity platform for developers](#) (formerly named Azure AD) v2.0. It enables you to acquire security tokens to call protected APIs. It uses industry standard OAuth2 and OpenID Connect.

Downloads

Full stats →

Total **1.9B**

Current version **20.1K**

Per day average **548.5K**

About

Last updated 3 days ago

Project website

Source repository

MIT license

Download package (4.02 MB)

Download symbols (1.27 MB)

Open in NuGet Package Explorer

MSAL controls

Microsoft Identity Client

<https://www.nuget.org/packages/Microsoft.Identity.Client/>

Extract `lib\net8.0\Microsoft.Identity.Client.dll` to your assemblies folder

Microsoft IdentityModel Abstractions

<https://www.nuget.org/packages/Microsoft.IdentityModel.Abstractions/>

Extract `lib\net8.0\Microsoft.IdentityModel.Abstractions.dll` to your assemblies folder

MSAL dotnetcore

%DLC%\bin\Progress.clrbridge.netcore.runtimeconfig.json

```
1  {
2      "runtimeOptions": {
3          "tfm": "net8.0",
4          "frameworks": [
5              {
6                  "name": "Microsoft.NETCore.App",
7                  "version": "8.0.0"
8              },
9              {
10                 "name": "Microsoft.WindowsDesktop.App",
11                 "version": "8.0.0"
12             }
13         ],
14         "configProperties": {
15             "System.Reflection.Metadata.MetadataUpdater.IsSupported": false
16         }
17     }
18 }
19
```

MSAL

Update .config in %DLC%\bin\

- `prowin.exe.config`
- `prowc.exe.config`
- `prowin32.exe.config (?)`

```
<configuration>
<startup useLegacyV2RuntimeActivationPolicy="true">
  <supportedRuntime version="v4.0"/>
</startup>
<runtime>
  <loadFromRemoteSources enabled="false" />
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1" >
    <!-- DO NOT REMOVE THIS probing ELEMENT -->
    <probing privatePath="Infragistics\winforms;telerik\winforms;dotnet" />
    <!-- START REDIRECTIONS -->
    <!-- for MSAL -->
    <dependentAssembly>
      <assemblyIdentity name="Microsoft.IdentityModel.Abstractions" publicKeyToken="31bf3856ad364e35" processorArchitecture="x86" />
      <publisherPolicy apply="no"/>
      <bindingRedirect oldVersion="6.35.0.0" newVersion="8.3.1.0"/>
    </dependentAssembly>
    <!-- END REDIRECTIONS -->
  </assemblyBinding>
</runtime>
</configuration>
```

MSAL controls

- Update assemblies.xml accordingly

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<references>
  <assembly name="System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
  <assembly name="Microsoft.Identity.Client, Version=4.74.1.0, Culture=neutral, PublicKeyToken=0a613f4dd989e8ae"/>
  <assembly name="Microsoft.IdentityModel.Abstractions, Version=8.3.1.0, Culture=neutral, PublicKeyToken=31bf3856ad364635"/>
</references>
```

Aquire JWT from Entra using MSAL

```
ServicePointManager:SecurityProtocol = SecurityProtocolType:Tls12.
```

```
publicApp = PublicClientApplicationBuilder  
    :Create(clientId)  
    :WithAuthority(AzureCloudInstance:AzurePublic, tenantId, true)  
    :WithRedirectUri(callbackUrl)  
    :Build()  
    .
```

```
authResult = publicApp:AcquireTokenInteractive("<clientId    :WithPrompt(Prompt:NoPrompt)  
    :ExecuteAsync()  
    :GetAwaiter()  
    :GetResult()  
    .
```

```
message "JWT from Azure Entra " authResult:AccessToken  
view-as alert-box.
```

Refresh JWT with Entra

```
ServicePointManager:SecurityProtocol = SecurityProtocolType:Tls12.
```

```
publicApp = PublicClientApplicationBuilder  
    :Create(clientId)  
    :WithAuthority(AzureCloudInstance:AzurePublic, tenantId, true)  
    :WithRedirectUri(callbackUrl)  
    :Build()  
    .
```

```
authResult = publicApp:AcquireTokenSilent(scopes, authResult?:account)  
    :WithForceRefresh(true)  
    :ExecuteAsync()  
    :GetAwaiter()  
    :GetResult()  
    .
```

```
message "JWT from Azure Entra " authResult:AccessToken  
view-as alert-box.
```

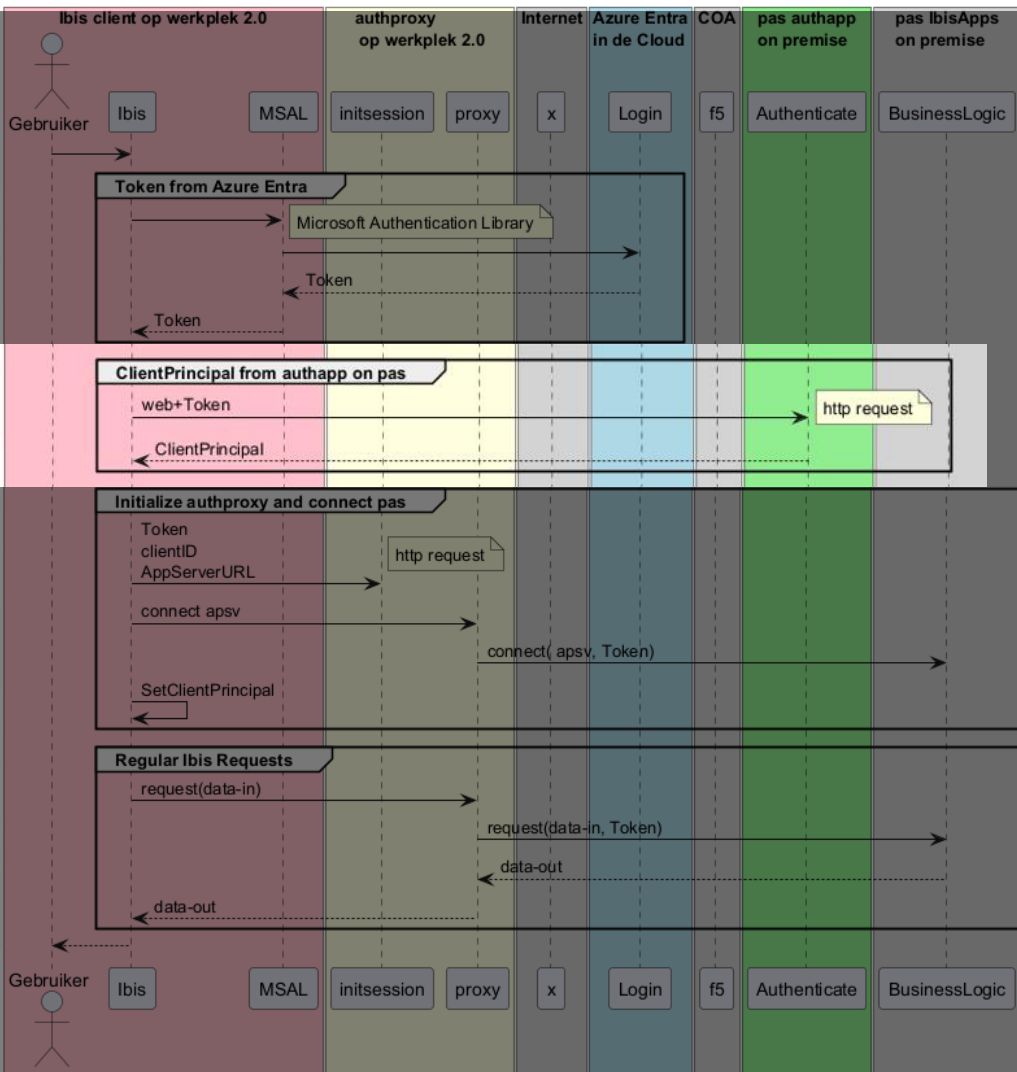
New login flow

- Azure Entra SSO
- Receive JWT

- Fetch ClientPrincipal

- Setup proxy
- Connect proxy
- SetClientPrincipal

- Ready



PASOE ablapp

- add a new ablapp on existing pasoe
- only configure web protocol
- have a single webhandler
- only accept GET
- configure security for jwt

PASOE security

update oeablSecurity.properties

```
<pasoe>/webapps/<ablApp>/WEB-INF/oeablSecurity.properties
```

```
client.login.model=oauth2
```

```
jwtToken.keystore.jwkurl=https://login.microsoftonline.com/<tenantId>/discovery/v2.0/keys
```

```
jwtToken.usernameField=onpremisesamaccountname
```

```
jwtToken.scopeNameField=roles
```

```
oauth2.resSvc.tokenServices=jwt
```

```
oauth2.resSvc.audience=<clientId>
```

PASOE debugging

configure logging logging.xml

```
<pasoe>/webapps/<ablApp>/WEB-INF/logging.xml
```

```
<!-- You may include logback <logger>....</logger> directives here in the event  
      that logging can only be turned on for specific Tomcat web application contexts.  
      For debugging security settings uncomment the following lines
```

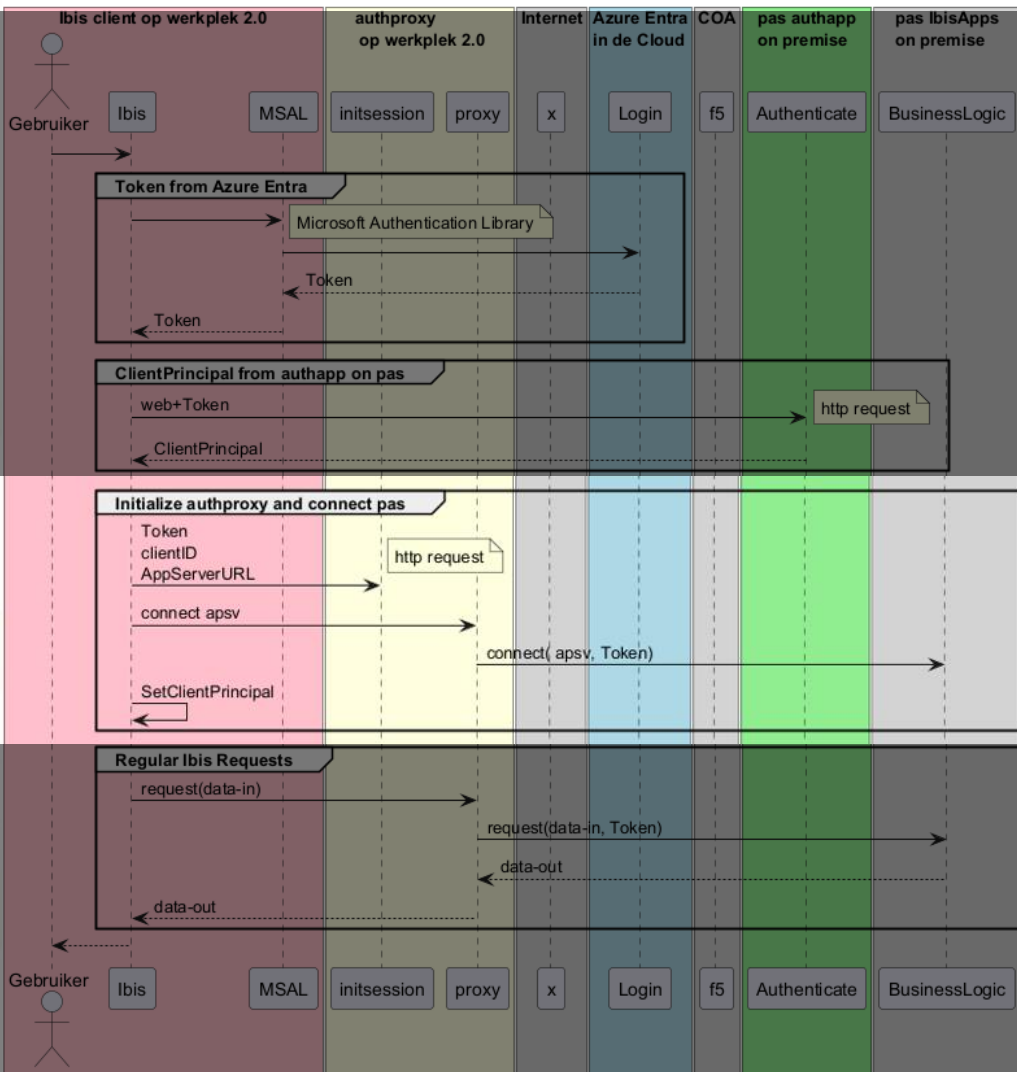
```
    <logger name="org.springframework.security.oauth2" level="TRACE" />  
    <logger name="org.springframework.security.jwt" level="TRACE" />  
    <logger name="org.springframework" level="TRACE"/>  
    <logger name="com.progress.appserv.services.security" level="TRACE"/>  
-->
```

Client principal service

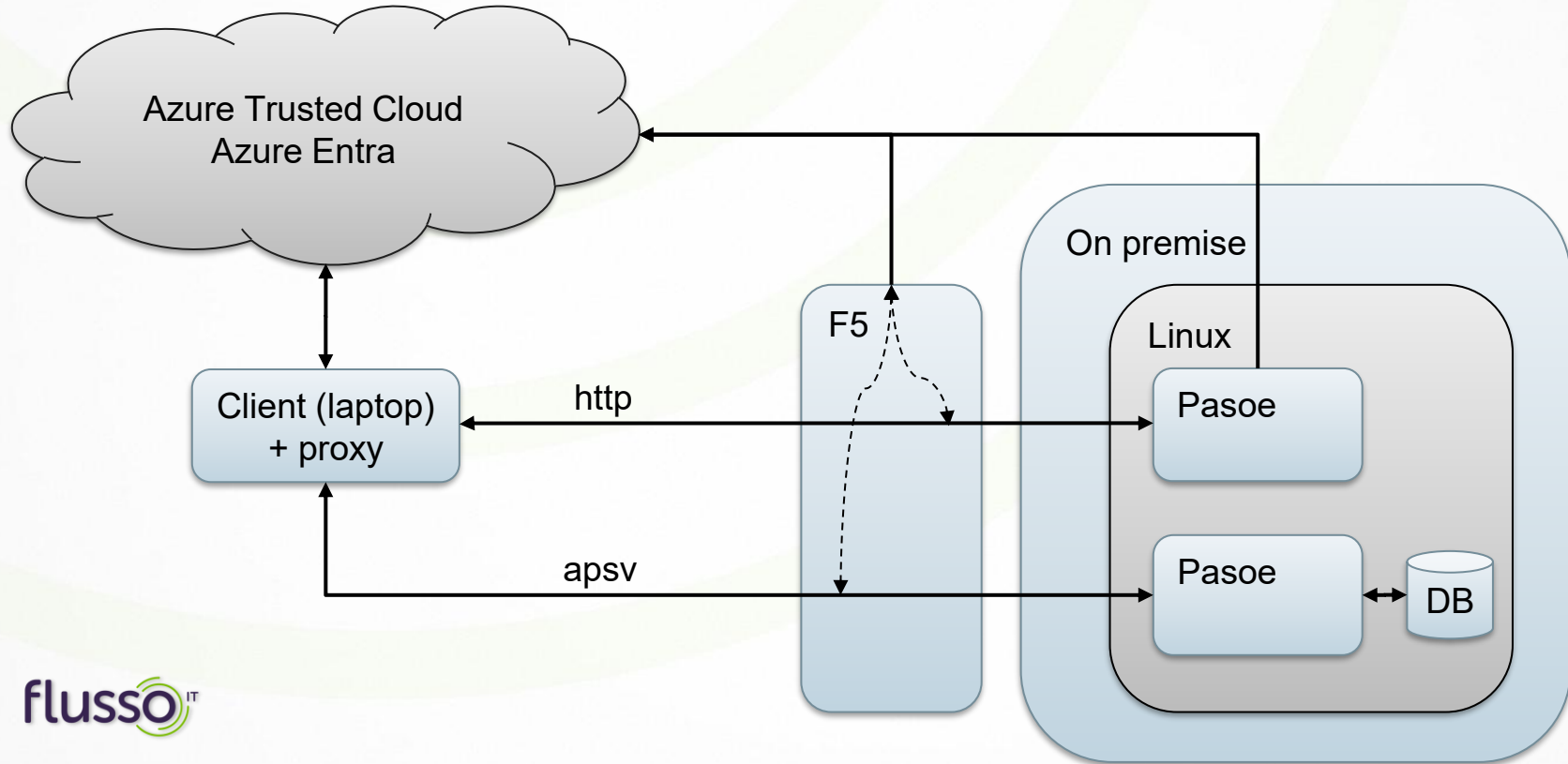
- OpenEdge converts jwt to client-principal
jwt claims -> client principal properties
- Create a IBIS specific client-principal
- return client-principal in base64

New login flow

- Azure Entra SSO
- Receive JWT
- Fetch ClientPrincipal
- Setup proxy
- Connect proxy
- SetClientPrincipal
- Ready



Zero Trust architecture

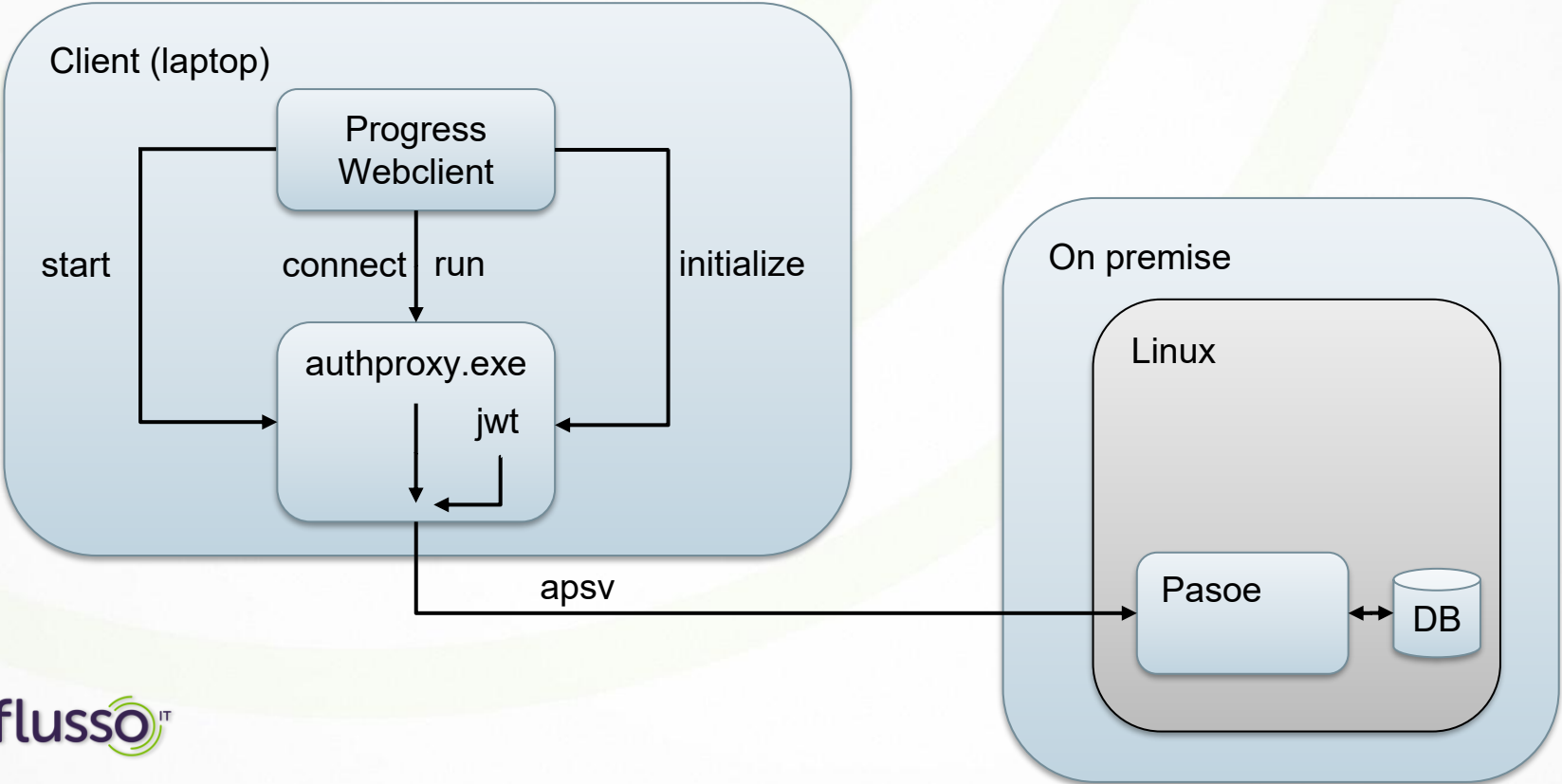


Authorization Proxy

Create a proxy between OpenEdge client and pasoe

- Runs on the client
- Started by the IBIS client
- Receives a JWT
- Connected by IBIS client as if it is an appserver
- Inserts Authorization header to each outgoing http request

Authorization Proxy



Authorization Proxy Start

```
authproxy.exe --nohostverify --proxyport 18820 --loglevel trace
```

Command line arguments

- proxyport (default=18810)
- nohostverify
- loglevel (default= info)
- authscheme (default= Bearer)
- version

Authorization Proxy Endpoints

POST	/quit	shutdown the proxy
GET	/alive	returns 200 OK
GET	/stats/{sessionId}	statistics
POST	/stats/ /{sessionId}	reset statistics
POST	/loglevel	set (body: {"loglevel": "debug"})
POST	/initsession	initialize for appserver connection
POST	/apsv	used by OpenEdge

Authorization Proxy Initialize

OpenEdge pasoe connection string

```
-URL https://backend.server/pasoe/apsv -sessionModel Session-free
```

OpenEdge http request

```
POST http://localhost:18810/initsession  
Content-Type: application/json
```

```
{ "sessionid": "12345",  
  "token": "jwt_recieved_from_entra",  
  "appserver": "https://backend.server/pasoe/apsv" }
```

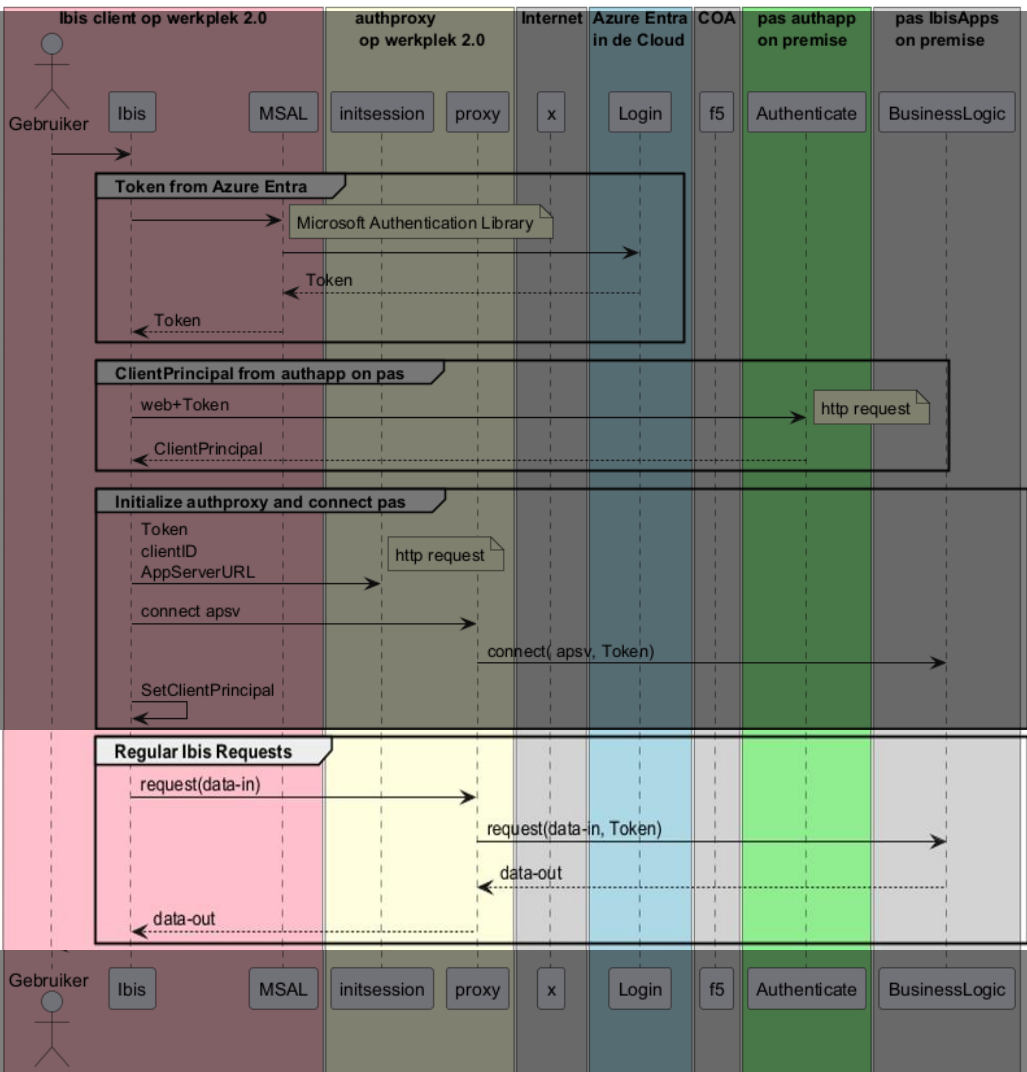
OpenEdge proxy connection string

```
-URL http://localhost:18810/apsv?AppService=12345 -sessionModel Session-free
```

New login flow

- Azure Entra SSO
- Receive JWT
- Fetch ClientPrincipal
- Setup proxy
- Connect proxy
- SetClientPrincipal

- Ready



Demo

Conclusion

To comply to zero trust is possible

But it is not trivial

- DIY
- reverse engineer APSV

Questions

- ?