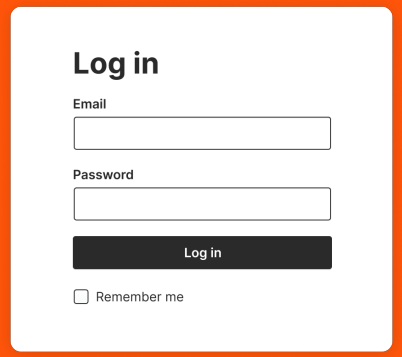
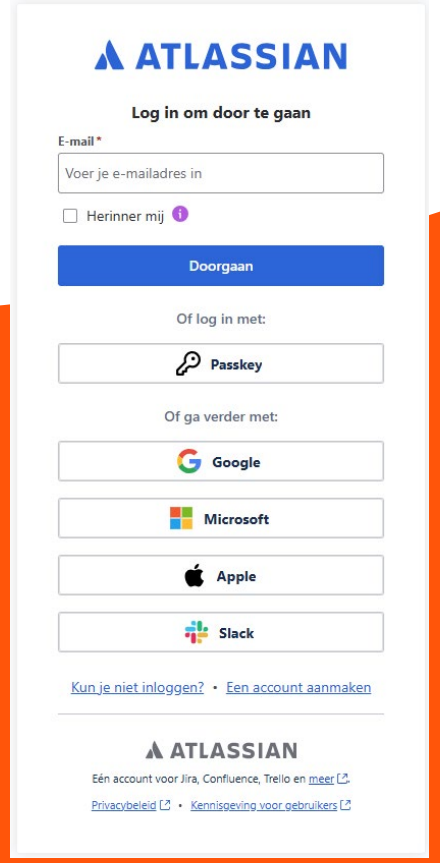
MODERN SECURITY IN DATAFLEX: OAUTH, SSO AND JWT

JULIAN VAN VEEN

TRADITIONAL LOGIN





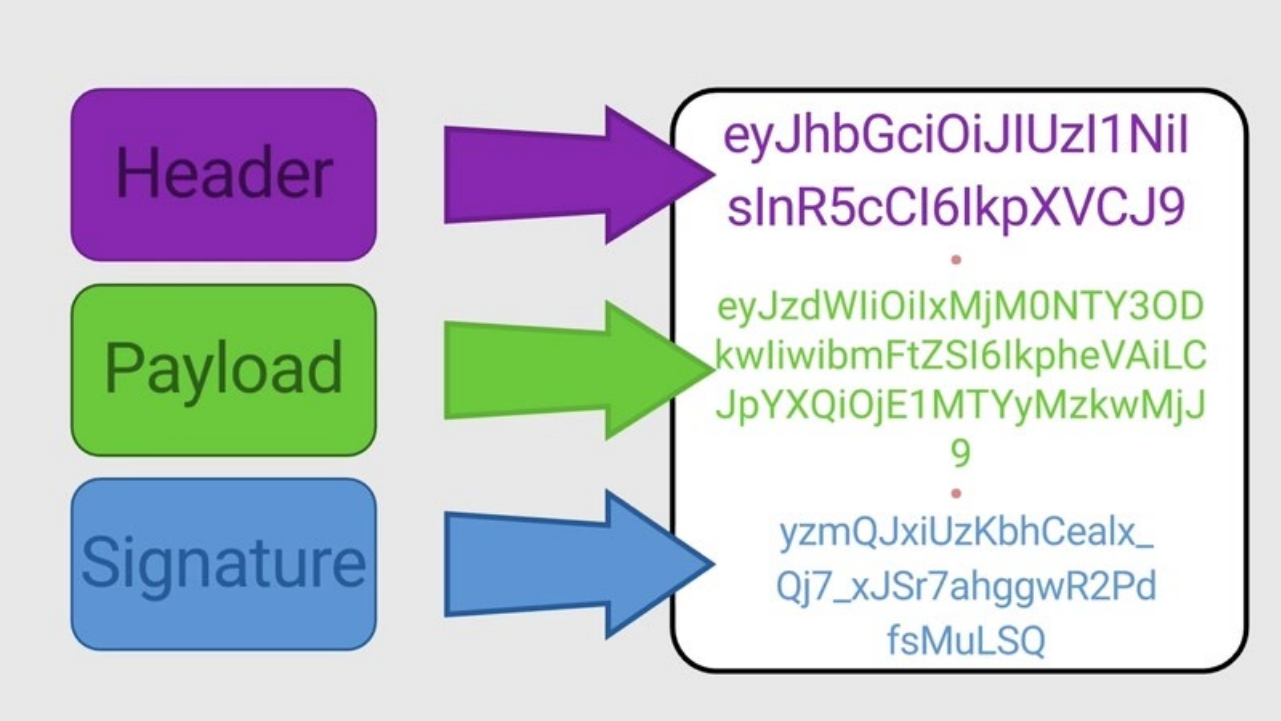
WHAT YOU WILL LEARN
SECURE LOGIN IN DATAFLEX

- JWT
- OAuth 2.0
- OpenID Connect
- Entra ID
- DataFlex as Identity server

JWT

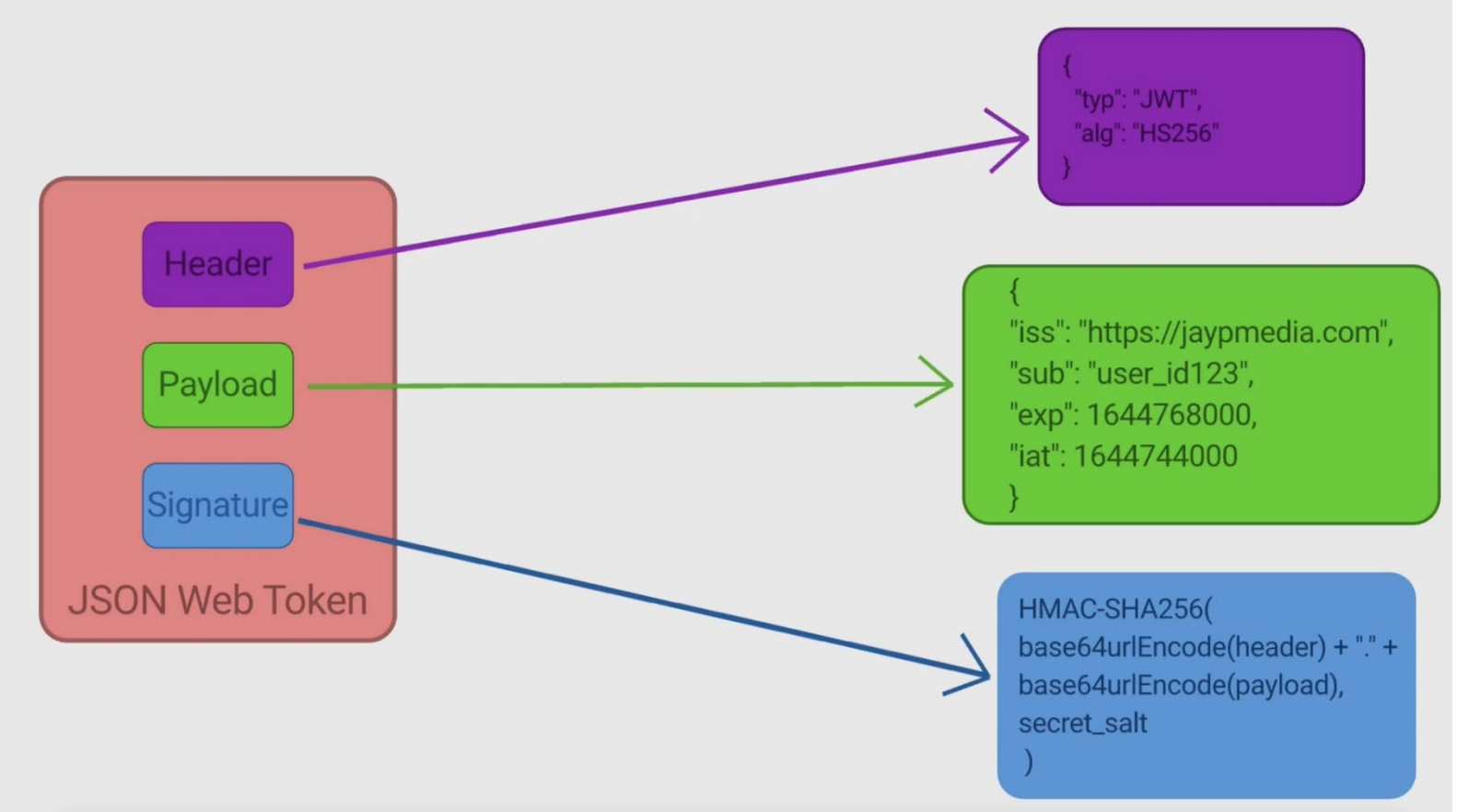
JWT
WHAT IS IT?

- What does JWT mean?
- What does it look like?
- What is it's purpose?



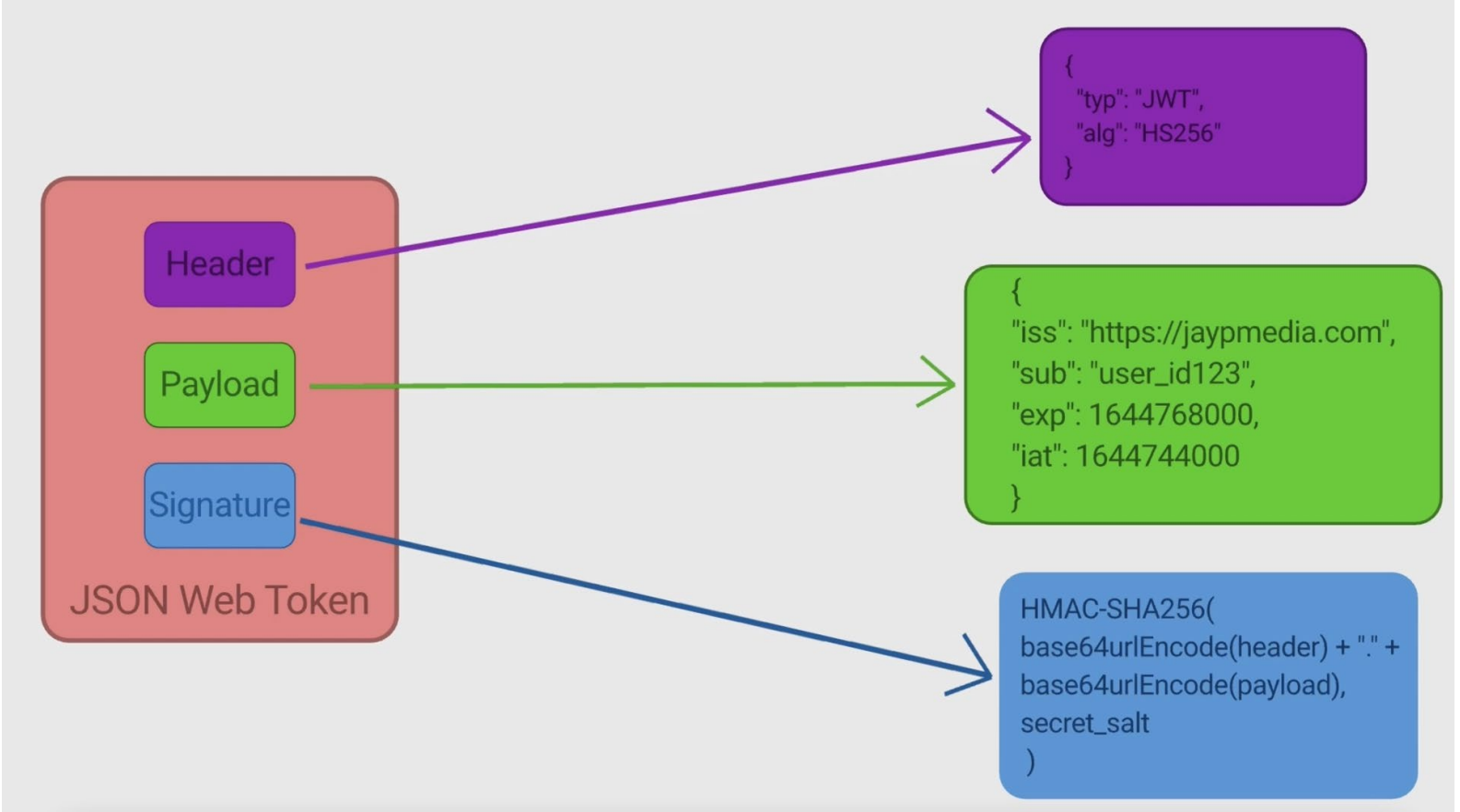
JWT
STRUCTURE

- Header
- Payload
- Signature
- xxx.yyy.zzz



JWT
CLAIMS

- iss
- sub
- exp
- iat
- aud
- nbf
- and more…

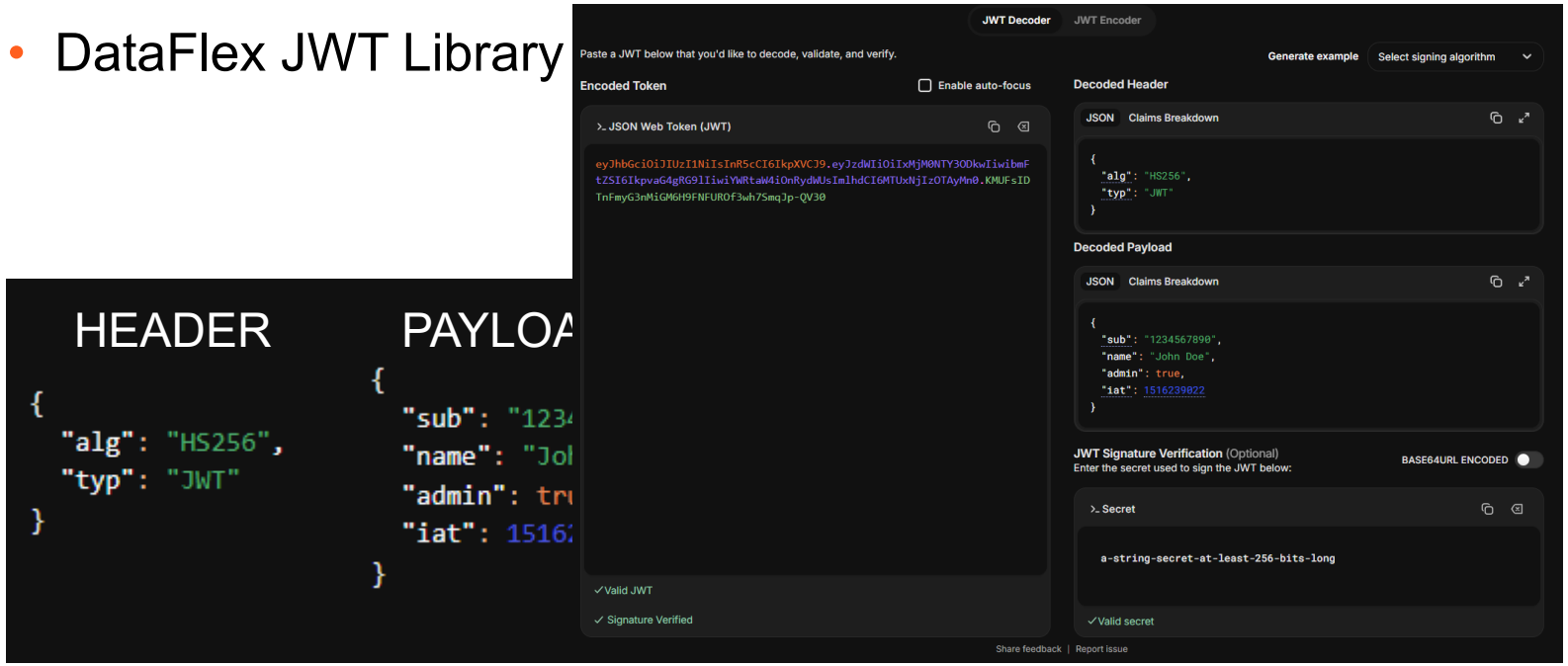


JWT
EXERCISE 1 – DECODE AND INSPECT A JWT

- Take the example JWT from the .docx (Word) file
- Do it manually to really grasp the technique
- For example use: CyberChef (cyberchef.siams.nl)

JWT
DECODE != TRUST

- Readable by default
- Always validate the signature (We can use the decoder on JWT.io i.e.)
- DataFlex JWT Library



JWT
SIGNING

- Private key -> Signature
- Public key -> Validate
- Most common: RSASSA-PKCS1-v1_5 + SHA256
- In the JWTs header this will be ‘RS256’ (R = RSA, S256 = SHA-256)

JWT
EXERCISE 2 – CREATE A PUBLIC/PRIVATE KEYPAIR USING OPENSSL

- For this exercise an openSSL installation is required.*
- *By default GIT installations includes an openSSL instance.

- To generate a private key:
  “**openssl genrsa –out private.pem 4096**”
- To extract the public key from the private key:
  “**openssl rsa –in private.pem –outform PEM –pubout –out public.pem**”
- Save these for later

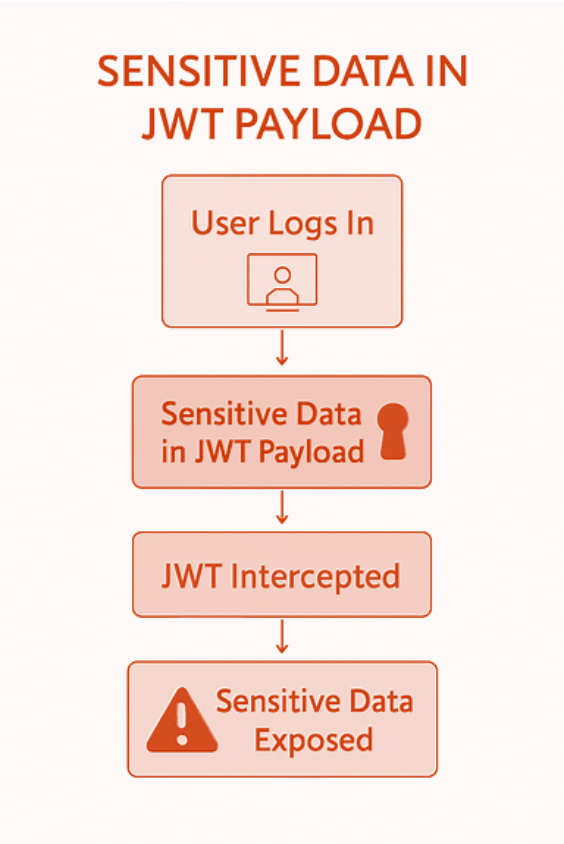
- Online solution at: cyberchef.siams.nl (Option is called “Generate RSA key pair”)

JWT
PROS/CONS

- Portable
- Stateless
- Tamper-resistant

- No sensitive data*
- Short lifetime

*Use JWE for sensitive data



JWT
DATAFLEX LIBRARY

- cJsonObject
- Public/Private keys
- Properties
- EncodeString
- L8w8jwt
- DecodeString

```
109     { Visibility=Private }
110     Function EncodeJWTEx Handle hoJsonWebToken String sKeyID String ByRef sJWT Returns Integer
111         /*
112         We need to make a variable for every JSON attribute because
113         the library requires memory addresses and pointers.
114         */
115         Integer iAlgorithm iSizeOfJWTResult iParsingResult iCounter iClaimsCounter iJsontype
116         Boolean bVoid bOk
117         String sPrivateKey sSub sIss sAud sJti sJWTResult sKIDHeaderName sMemberName sMemberValue
118         String[] saBlacklistMembers
119         UBigInt ubiIat
120         Pointer pJWTResult lpMemberName lpMemberValue lpWholeList lpIterate
121         UChar[] ucaResult
122         l8w8jwt_encoding_params stl8w8jwt_encoding_params
123         l8w8jwt_additional_claims stl8w8jwt_additional_header_claims
124         l8w8jwt_additional_claims stl8w8jwt_claim_iterate
125         l8w8jwt_additional_claims stl8w8jwt_additional_payload_claims
126         stJWTKey tJWTPrivateKey
127
128         If (hoJsonWebToken = 0) Function_Return C_JWT_INVALID_JSONWEBTOKEN_HANDLE
129
130         Get JWT_Encode_Init (AddressOf(stl8w8jwt_encoding_params)) to bVoid
131         //Alg
132         Get peAlgorithm of hoJsonWebToken to iAlgorithm
133         If (iAlgorithm < 0 or iAlgorithm > 11) Function_Return C_JWT_NOT_A_SUPPORTED_ALG
134         Move iAlgorithm to stl8w8jwt_encoding_params.alg
```

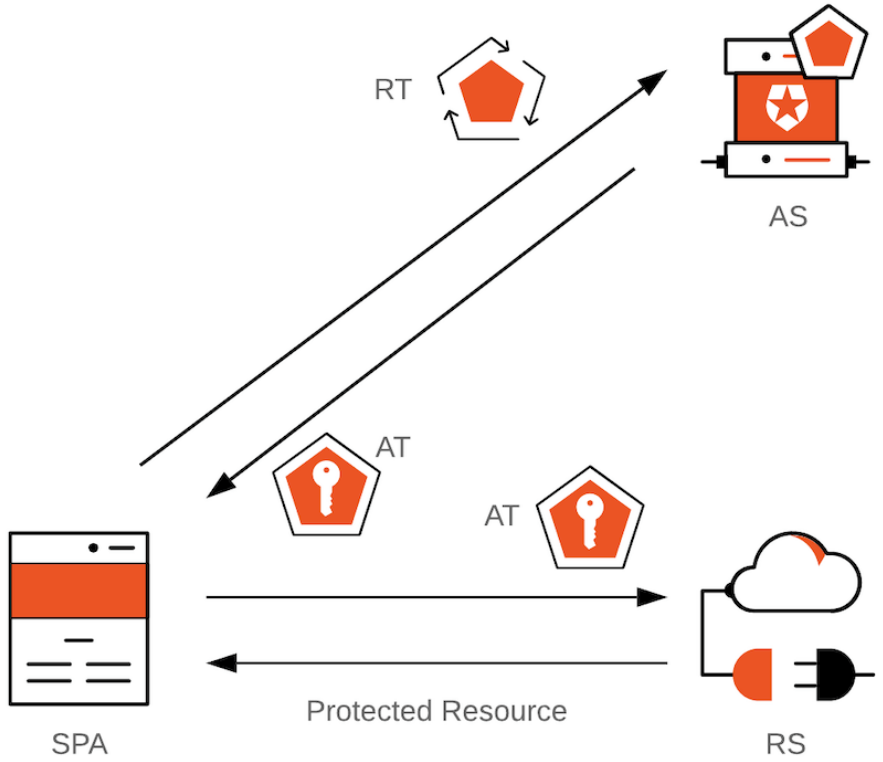
JWT
EXERCISE 3 – CREATE AND VALIDATE A JWT IN DATAFLEX

- Use the JWT library to create and validate a token
- Steps:
  - Pull the git repo from “DAE-JulianVanVeen/ModernSecurityFlexRoads2026 (or use the zip from bit.ly/MSFR2026)
  - Checkout the “Exercise 3 – Start” branch
  - Open the workspace (ModernSecurity.sws)
  - Create a new view to expirement on (File -> New -> Views… -> Data Entry)
  - Create a cJsonWebToken object inside your view
  - Add your private and public key from exercise 2 by calling ‘AddJWTPrivateKey’ and ‘AddJWTPublicKey’ (the calls can be made inside the object itself and the return value can be voided to gVoid)
  - Add a button to your view that will set some random properties of the JWT object (like JWTID, Issuer, Audience etc). After this call EncodeString and write the result to a readonly cWebEdit.
  - Restart the webapp in the debugger and create a button that calls DecodeString on the JWT object with the result of the previous step. Now get the property values and look at the result. (Place a breakpoint in the OnClick of the button)

OPENID CONNECT

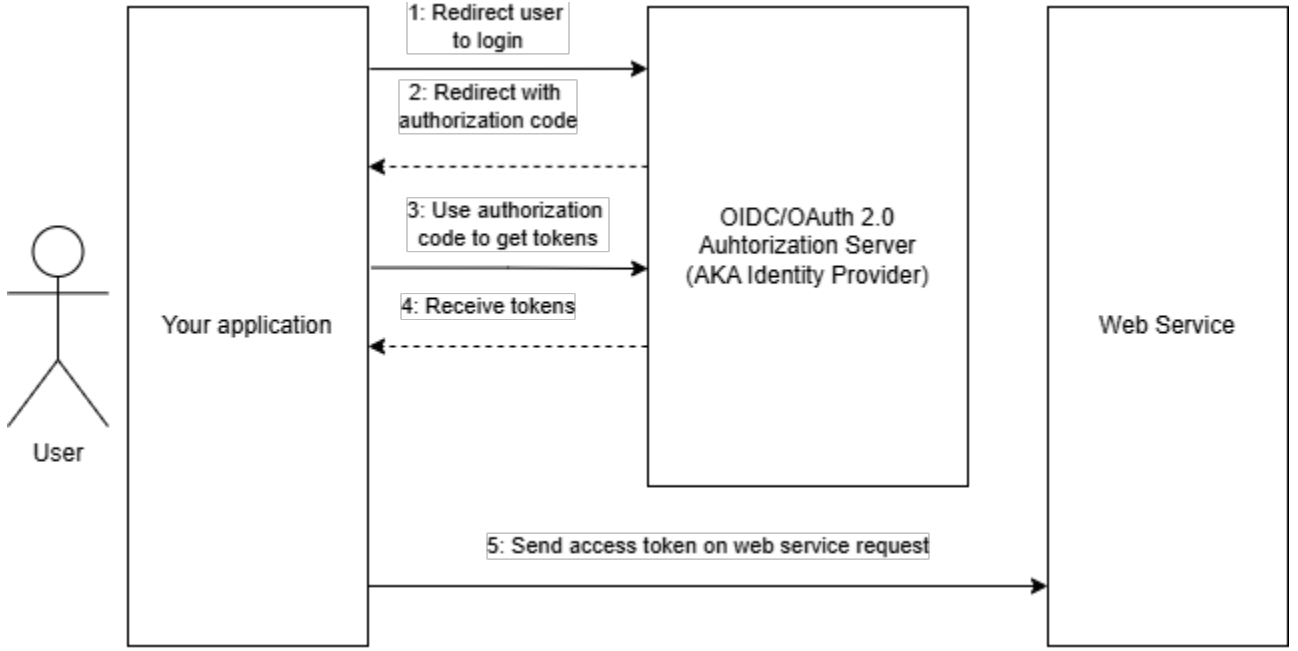
OAUTH/OIDC
TOKENS

- Authorization Server (AS)
- Single page applications (SPA)
- Resource Servers (RS)
- Authorization code
- Access Token (AT)
- Refresh Token (RT)
- ID Token (OIDC only)



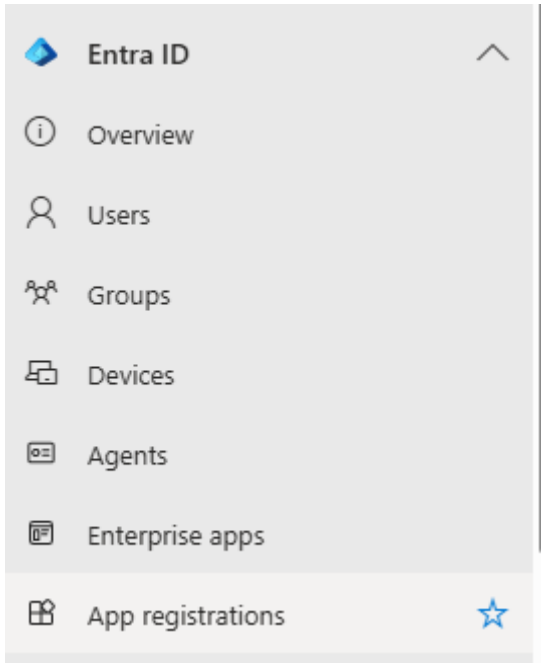
OAUTH/OIDC
AUTHORIZATION CODE FLOW

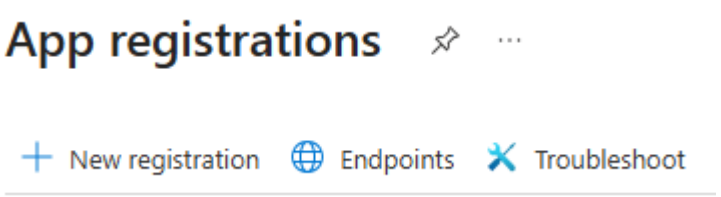
- Client Application
- Identity provider
- Callback
- Authorization code
- Tokens

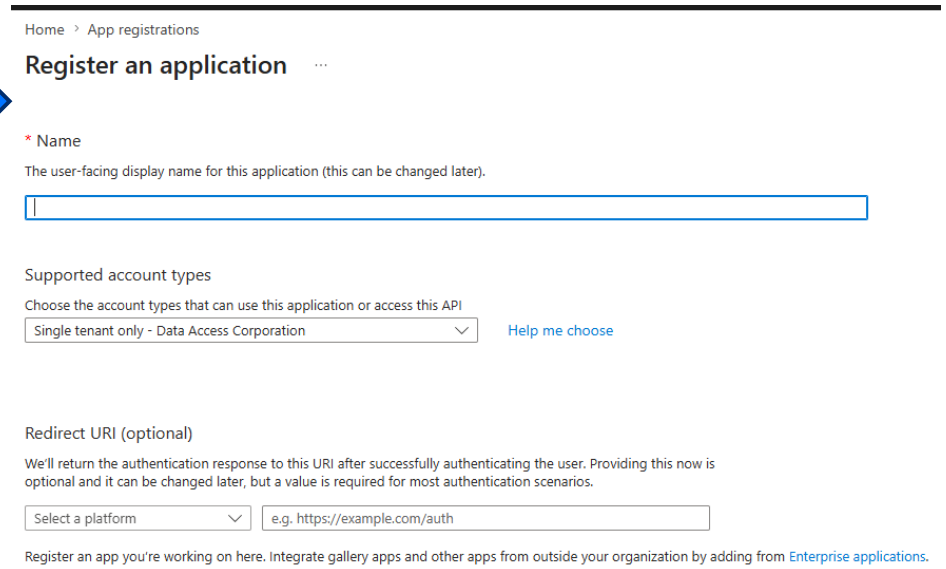


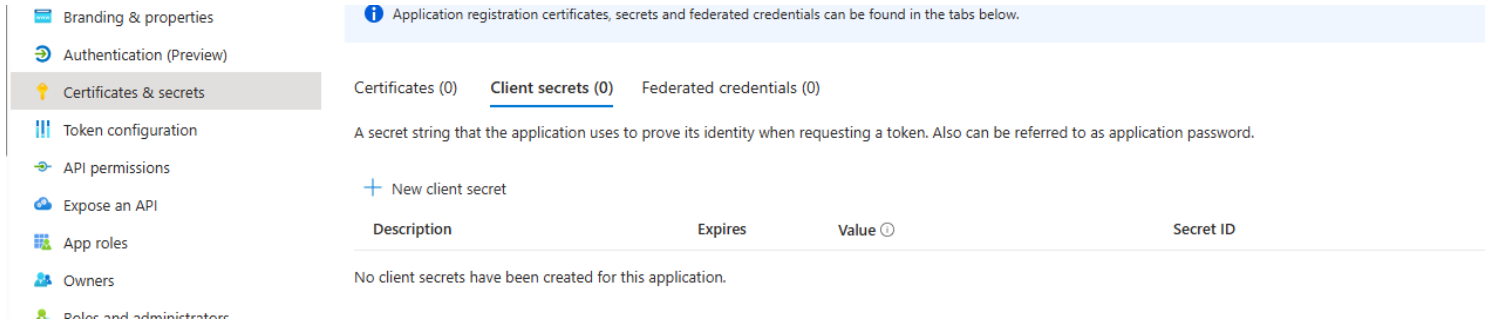
OAUTH/OIDC
EXERCISE 4 – REGISTER AN APPLICATION IN ENTRA ID

- App Registration
- Redirect URI









OAUTH/OIDC
DATAFLEX SSO LIBRARY

- cSsoApi (inside oWebApp)
  - Global handle
  - Manages callbacks
- Provider
  - Configuration
- CallbackHandler
  - Receives the callbacks
- cSsoOnload_mixin
- State object
  - Implements a mixin
  - Keeps track of SSO authentication state
  - Allows for multiple logged in identities at multiple different providers

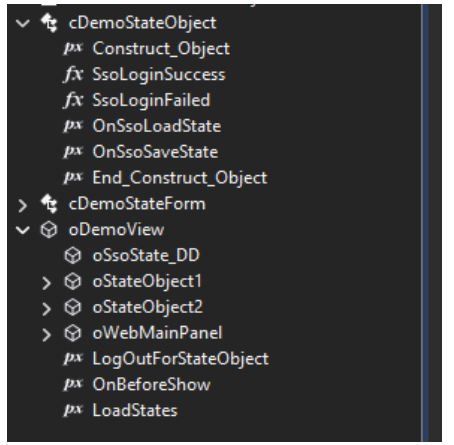
```
025     Object oSso is a cSsoApi
026         Set psHomeURL to "http://localhost/SSODemo"
027         Set psCallbackURLLogin  to "http://localhost/SSODemo/SSOCallback/Login"
028         Set psCallbackURLLogout to "http://localhost/SSODemo/SSOCallback/Logout"
029     End_Object
```

```
095     Object oEntraTestProvider is a cSsoProviderOpenId
096         Set peAlgorithm to C_JWT_ALG_RS_256
097         Set psProviderID to "ENTRA"
098         Set psScopeParam to "openid offline_access email profile"
099         Set psIdentityClientID to "61e901c2-89c1-491f-92ab-3f5b7ec62e23"
100         Set psIdentityClientSecret to "**********************************"
101         Set psSSOHost to "https://login.microsoftonline.com/"
102         Set psDiscoveryEndpoint to "30626ff8-9106-4545-9fd8-73794e98f118/.well-known/openid-configuration"
103         Set pbAutoDiscovery to True
104         Set pbAutoloadCertificates to True
105     End_Object
```

```
235     Use SessionManager.wo
236     Use Login.wo
237     Use WebResourceManager.wo
238
239     Use Dashboard.wo
240     Set phoDefaultView to oDashboard
241     Use SSO\CallbackHandler.wo
242     Use DemoView.wo
```

```
Use SSO\cSsoOnLoad_mixin.pkg
Import_Class_Protocol cSsoOnLoad_mixin

Procedure OnLoad
    Forward Send OnLoad
    Send SsoOnLoad
End_Procedure
```

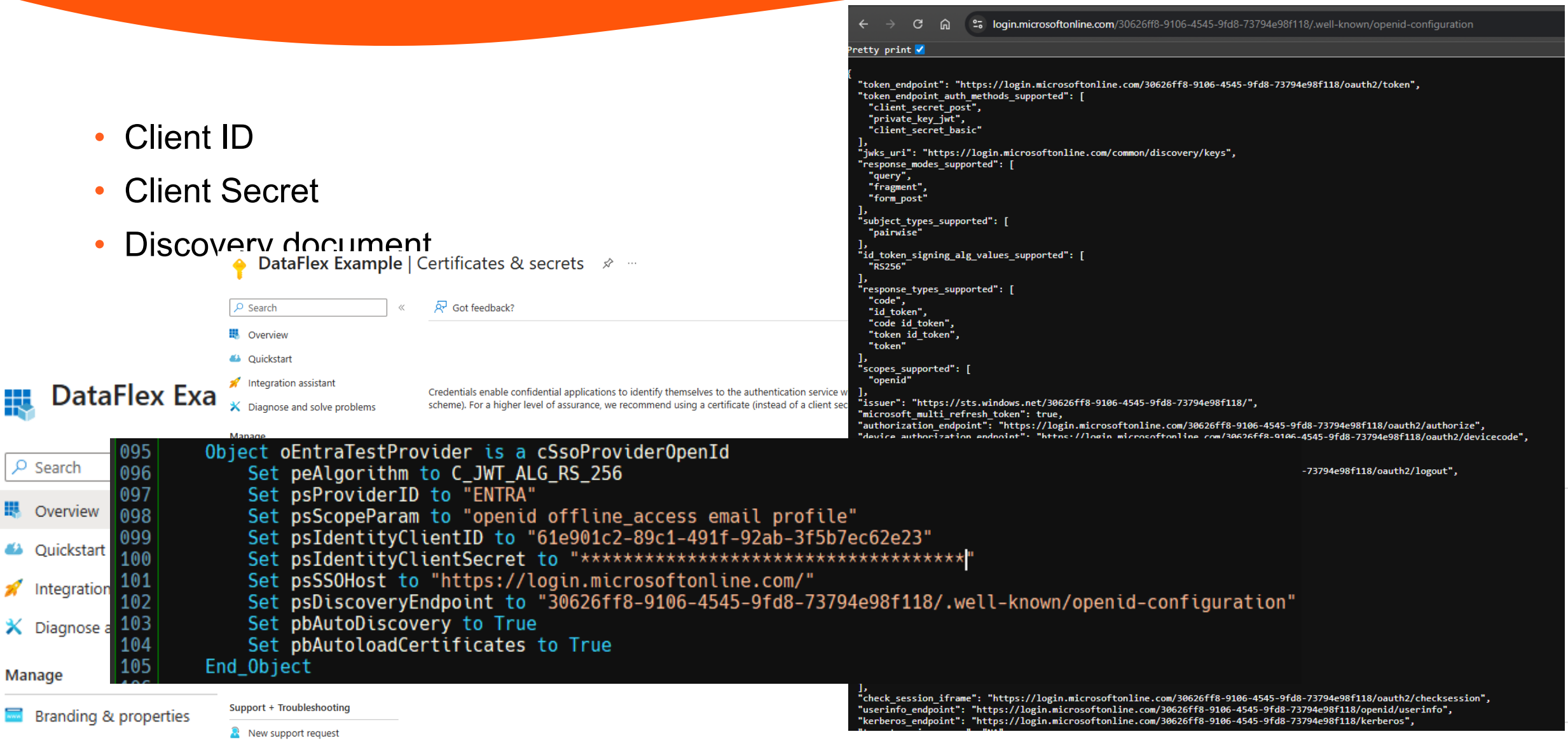
```
Class cSsoOnLoad_mixin.pkg is a Mixin
    Procedure SsoOnLoad
        String sError sUrl sStateHash

        // Check for an error and show
        Get UrlParameter C_SSO_LOGIN_ERROR_PARAMETER to sError

        If (SizeOfString(sError) > 0) Begin
            Send UserError sError "Login failed"
            // Remove the query parameters
            Get psHomeURL of ghoSso to sUrl
            Get StateHash of ghoWebApp to sStateHash
            Move (sUrl + "/#" + sStateHash) to sUrl
            Send HistoryReplaceState of ghoWebApp sUrl
        End
    End_Procedure
End_Class
```



/*50 YEARS*/

DATAFLEX SSO LIBRARY
PROVIDER SETUP

- Client ID
- Client Secret
- Discovery document



DATAFLEX SSO LIBRARY
STATE OBJECTS

- cSsoState_mixin
- Send IdentityLogin “provider_id”
- CallbackHandler.wo
- SsoLoginSuccess / SsoLoginFailed
- Important detail: cWebHttpHandler + WebProperties
- Session Manager (User mapping)

DATAFLEX SSO LIBRARY
EXERCISE 5 – CONFIGURE THE DATAFLEX SSO CLIENT LIBRARY

- Goal: Connect a DataFlex WebApp to Entra ID
- Steps:
  - Install the **Single Sign On** package
  - Setup the cSssoApi (inside the oWebApp object)
    - **psHomeUrl**, **psCallbackURLLogin** and **psCallbackURLLogout**
  - Setup a provider with information from **EntraID**. It is recommended to enable **pbAutoDiscovery** and **pbAutoloadCertificates** (disable this if you want to challenge yourself)
  - Include the **CallbackHandler.wo** inside the oWebApp object (where you normally put use statements for your views)
  - Handle SSO errors by using the **cSsoOnLoad_mixin** and then calling SsoOnLoad inside the WebApp's **OnLoad**.
  - Implement a simple SSO state object that has implementations for **OnSsoLoadState**, **OnSsoSaveState**, **SsoLoginSuccess** and **SsoLoginFailed**.
    - Tip 1: Create an empty view and make a class that implements the cSsoState_mixin, then create an object of that class inside the new view.
    - Tip 2: Use the embedded database to save SSO state to (it can't be put into webproperties because when callbacks come in the webprop store is not initialized).
    - Tip 3: When using the embedded database there is no DateTime(2) field, make clever use of the DataDictionary class to have two separate columns with types *Date* and *ASCII* (for the hh:mm:ss). See the WebAppSession DataDictionary.
  - To test the setup do: "**Send IdentityLogin of oYourObjName 'YOUR PROVIDER ID'"**

DATAFLEX SSO LIBRARY
EXERCISE 6 – MAP AN EXTERNAL USER TO A LOCAL WEBAPP USER

- Tip: Inspect and use the cSsoWebSessionManager.
- For this we will need to implement **OnSsoLoadState**, **OnSsoSaveState** and **MapSsoUserInfoToLocalUser**.
- Make sure to increase **WebAppUser.LoginName** to be 255 characters, this is the maximum index size of the embedded database. When changing the size **WebAppSession.LoginName** needs to be altered too.
- We will need to save the “mapping” so we can reconnect a third party identity to the local WebAppUser record.
- Don’t forget to set the **psStateKey** property so callbacks can be routed back accordingly.
- Bonus: Add an extra button on the login screen that does “**Send IdentityLogin of ghoWebSessionManager ‘YOUR PROVIDER ID’**”. This will enable SSO logins to your webapp.

DATAFLEX SSO LIBRARY
EXCERCISE 7 – UNDERSTANDING REFRESH TOKEN ROTATION

- Scope ‘offline_access’
- Use the RefreshTokens function of the SSO library and the debugger for this exercise
- Tip 1: Make use of “Set next instruction” during debugging to skip certain parts of code (i.e. when the library thinks your access token is still valid so a refresh is unnecessary)
- Tip 2: Copy an old refresh token before the first refresh or manually mark it as “used” in the database.

DATAFLEX OAUTH SERVER LIBRARY

DataFlex

/*50 YEARS*/

OAUTH SERVER LIB
PREREQUISITES

- Login view
- Consent view
- Keystore
- Provider
- Handler
- Well-known handler

```
Object oOidcHandler is a cOidcHandler
    Procedure Setup
        Handle hoLoginView hoConsentView

        Set phoOidcProvider to oOidcProvider

        Get GetLoginView of ghoWebApp to hoLoginView
        If (hoLoginView <> C_WebUnresolvedObject) Begin
            Set phoLoginView to hoLoginView
            Set phoOidcProvider of hoLoginView to oOidcProvider
            Set
        End
        Else Beg
            Erro
        End

        Get phoO
        If (hoConsentView <> C_WebUnresolvedObject) Begin
            Set phoOidcConsentView to hoConsentView
            Set phoOidcProvider of hoConsentView to oOidcProvider
            Set phoOidcHandler of hoConsentView to Self
            Send RegisterNavigateForwardPath ntNavigateBegin hoConsentView ghoWebApp ""
        End
        Else Begin
            Error DFERR_PROGRAM "No consent view setup for OAuth"
        End
    End_Procedure
```

```
Use Oidc\cOidcWellKnownHandler.pkg

Object oOidcWellKnown is a cOidcWellKnownHandler
    Set phoOidcProvider to oOidcProvider
    Set phoOidcKeyStore to oOidcKeyStore
    Set phoOidcHandler  to oOidcHandler
End_Object
```

OAUTH SERVER LIB
KEEP IT SIMPLE

- Demo purposes
- KISS = Keep It Stupid Simple!
- Use the demo code (See GitHub)
- Use the instructions on the handout

DATAFLEX OAUTH SERVER LIBRARY
EXERCISE 8 – SETUP A DATAFLEX WEBAPP AS YOUR VERY OWN IDENTITY SERVER

- Goal: Understand how a DataFlex WebApp can act as an identity server
- Tip: Look at the demo available in the training zip
- Steps:
  - Include the login and consent views
  - Add a property for the consent view to the WebApp object (Property Handle phoOidcConsentView 0)
  - Configure a keystore and include it in the WebApp as use statement (separate pkg file)
  - Configure a provider (implement the events with hardcoded data or use the default provider class with the included SQL script to generate the default tables. Look at the example code in chapter 8.8 of the handout)
  - Set the property ‘psApplicationBaseUrl’ of the provider to your webapps base url
  - Configure a cOidcHandler (look at Oidc\cOidcDefaultHandler
  - Configure and include the well-known handler (cOidcWellKnownHandler)
  - Now test your identity server by crafting an authorize URL
    - The url should look something like: **http://localhost/ApplicationName/oidc/v1/authorize?clientid=xxx&scope=openid+profile+email&response_type=code&redirect_uri=http%3A%2F%2Flocalhost%2Fmy%2Dcallback**

/* THAT'S IT! YOU ARE NOW AUTHORIZED TO ASK QUESTIONS. :D*/

Enjoy the rest of FlexRoads!