



MODERN SECURITY IN DATAFLEX: OAUTH, SSO AND JWT

JULIAN VAN VEEN

TRADITIONAL LOGIN

Log in

Email


Password

Remember me

ATLASSIAN

Log in om door te gaan

E-mail *

Herinner mij 

Of log in met:

Of ga verder met:

[Kun je niet inloggen?](#) • [Een account aanmaken](#)

ATLASSIAN

Eén account voor Jira, Confluence, Trello en [meer](#)

[Privacybeleid](#) • [Kennissegeving voor gebruikers](#)

WHAT YOU WILL LEARN

SECURE LOGIN IN DATAFLEX

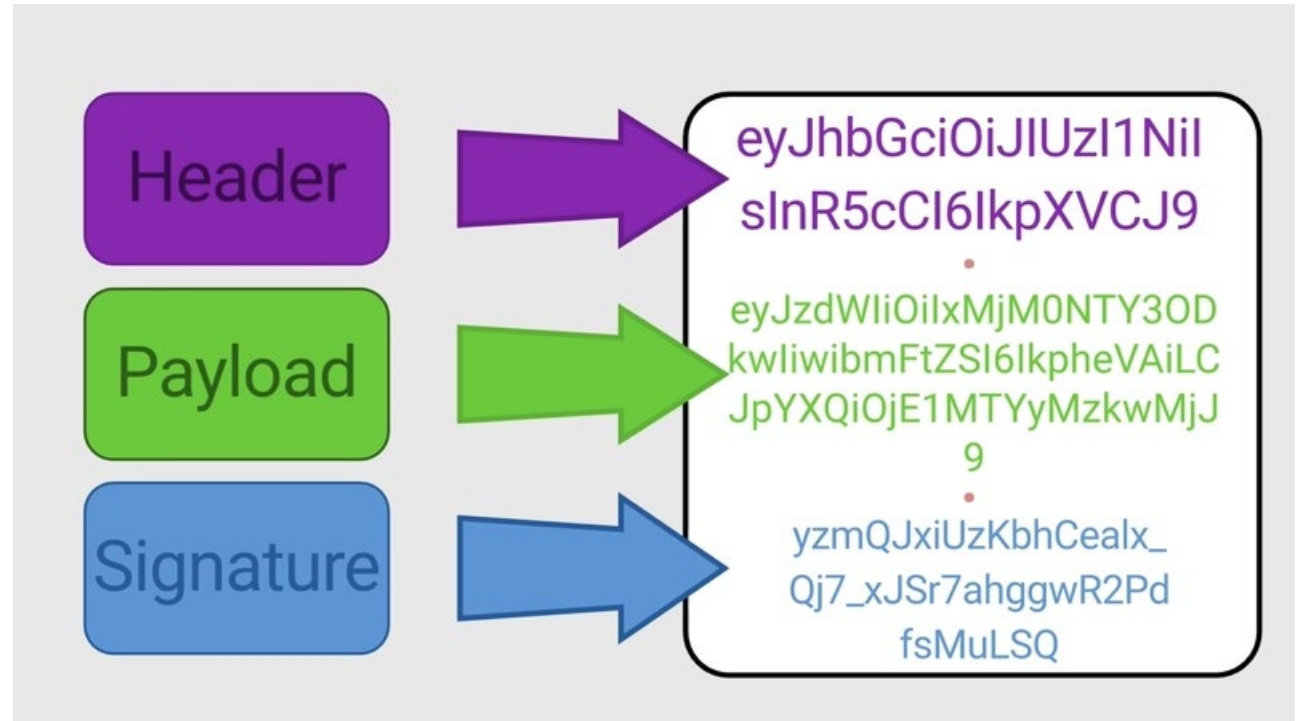
- JWT
- OAuth 2.0
- OpenID Connect
- Entra ID
- DataFlex as Identity server

JWT

JWT

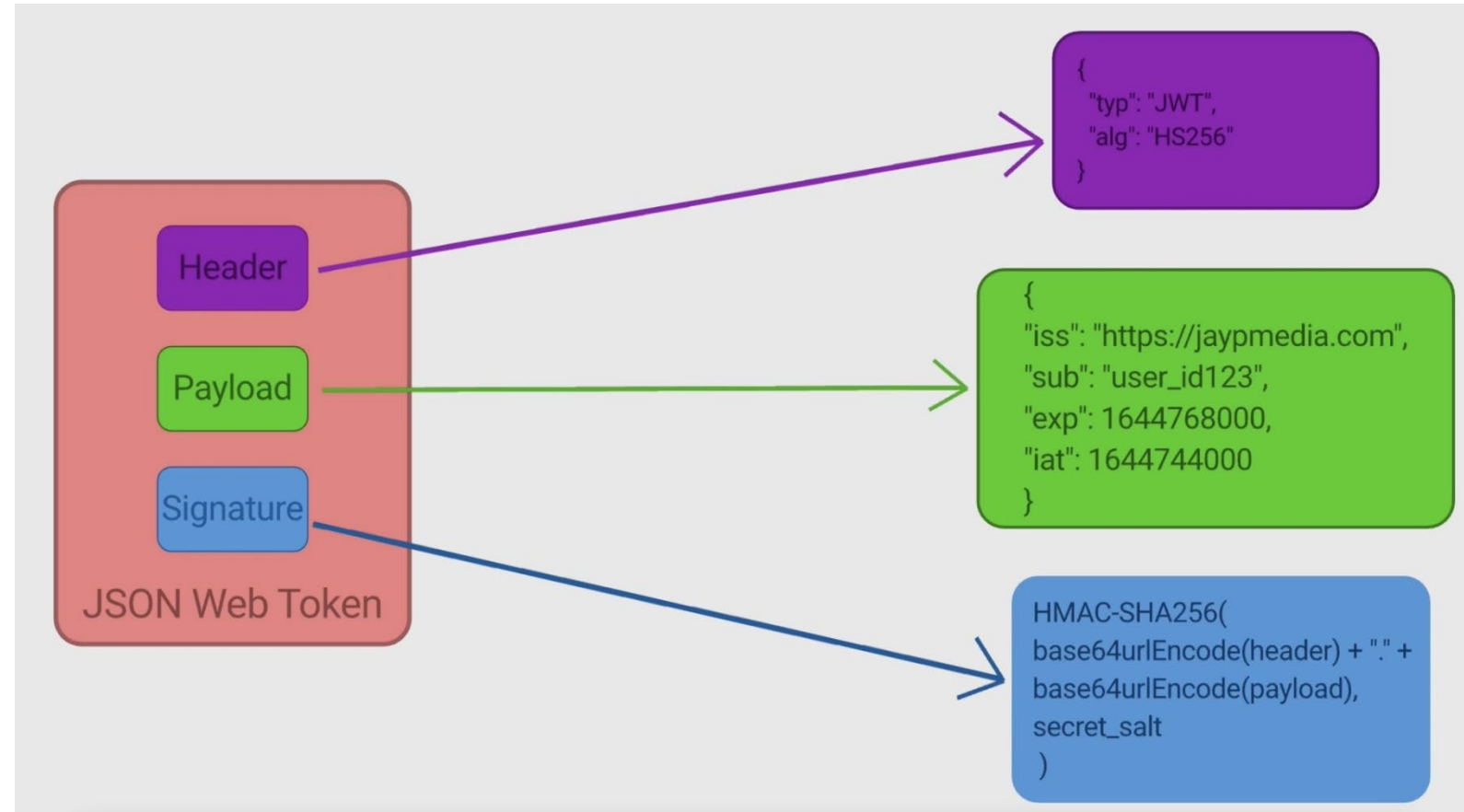
WHAT IS IT?

- What does JWT mean?
- What does it look like?
- What is its purpose?



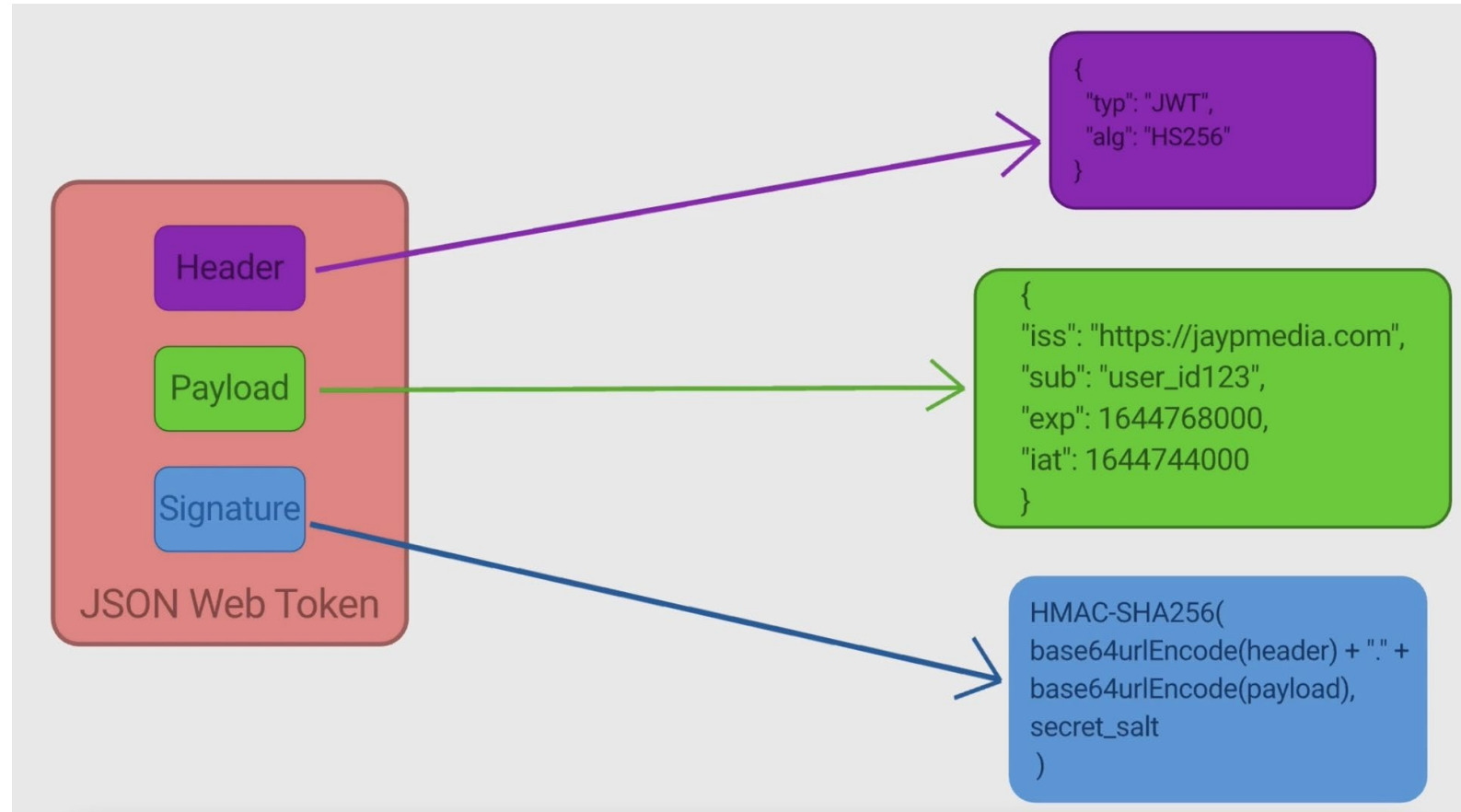
JWT STRUCTURE

- 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)

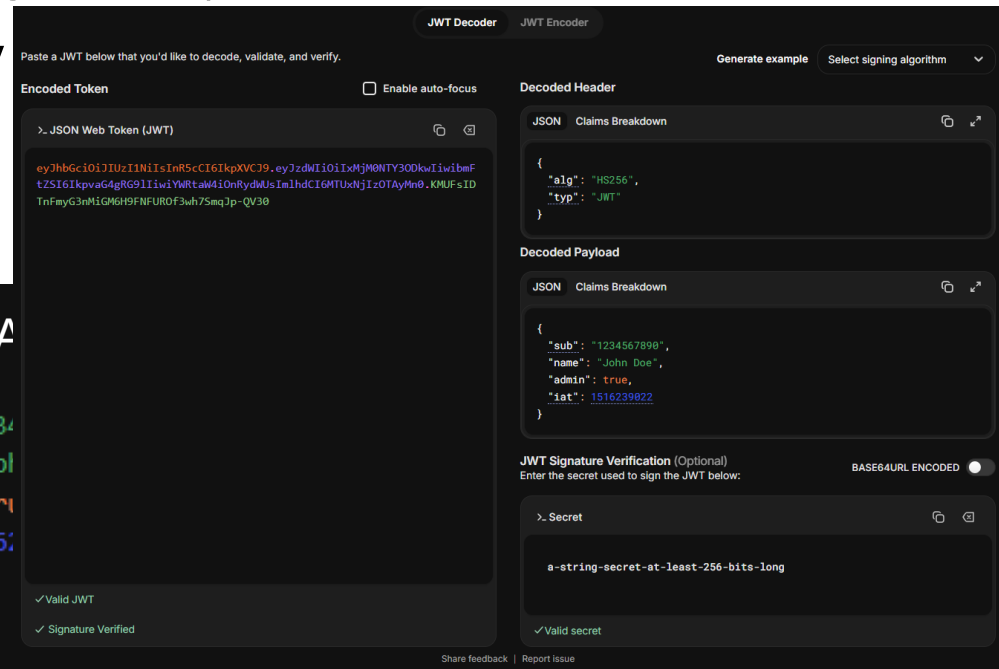


The screenshot shows the CyberChef web interface. The 'Input' section contains two lines of JWT tokens: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9` and `eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoiYWRtaW4iOnRydWUsIm1hdCI6MTUxNjIzOTAyMn0`. The 'Output' section shows the decoded JSON object: `{"alg": "HS256", "typ": "JWT"}{"sub": "1234567890", "name": "John Doe", "admin": true, "iat": 1516239022}`. The interface includes file management icons and a 'Raw Bytes' view option.

JWT DECODE != TRUST

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

```
HEADER      PAYLOAD
{           {
  "alg": "HS256",
  "typ": "JWT"
}           {
  "sub": "1234",
  "name": "John Doe",
  "admin": true,
  "iat": 1516239822
}           }
```



JWT SIGNING

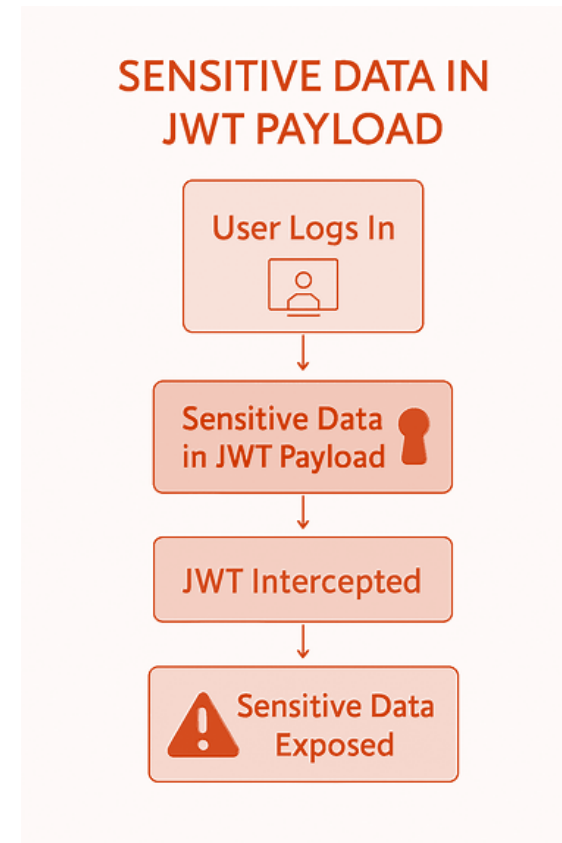
- Private key -> Signature
- Public key -> Validate
- Most common: RSASSA-PKCS1-v1_5 + SHA256
- In the JWTs header this will be 'RS256'
(R = RSA, S256 = SHA-256)

- 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

```
108
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
135
```

JWT

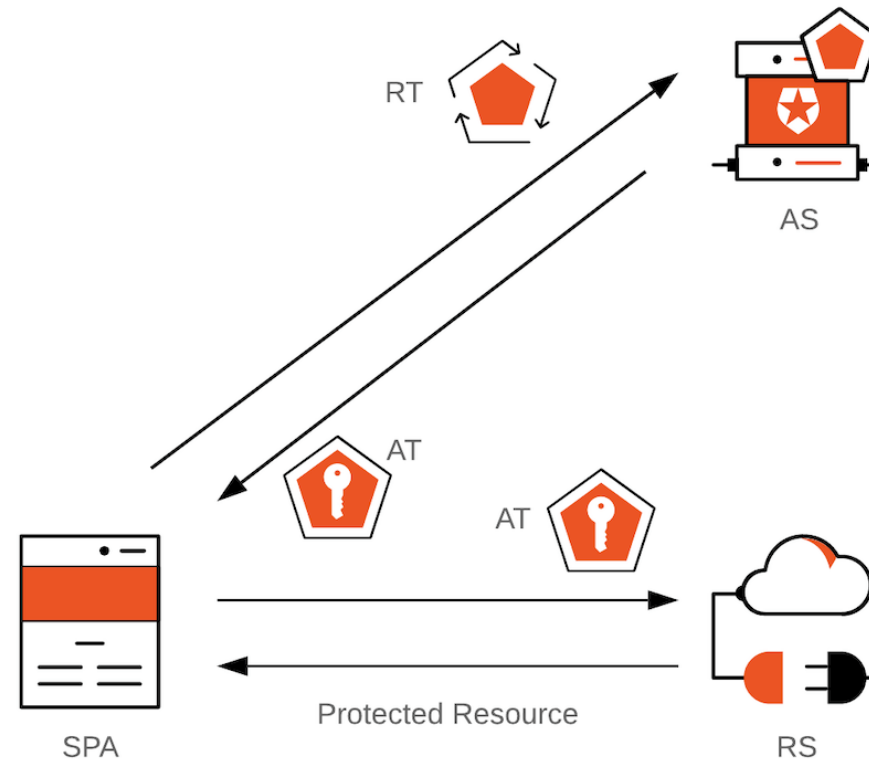
EXERCISE 3 – CREATE AND VALIDATE A JWT IN DATAFLEX

- 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 experiment 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

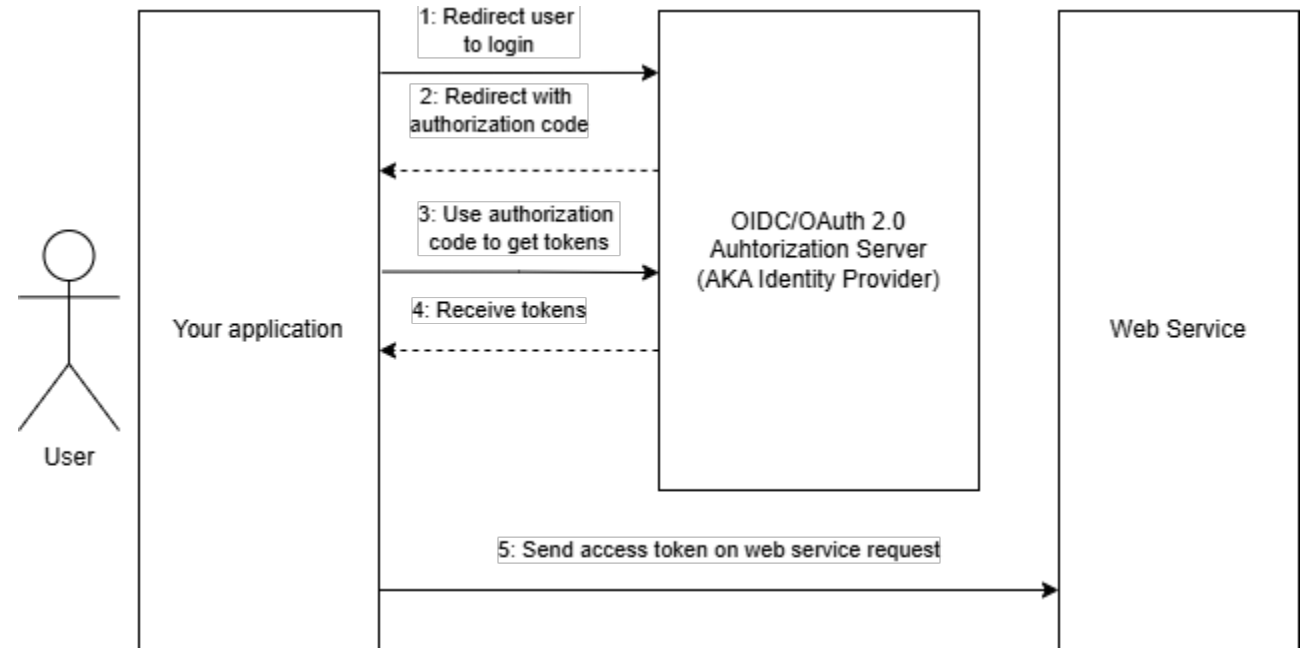
OAuth/OIDC TOKENS

- 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

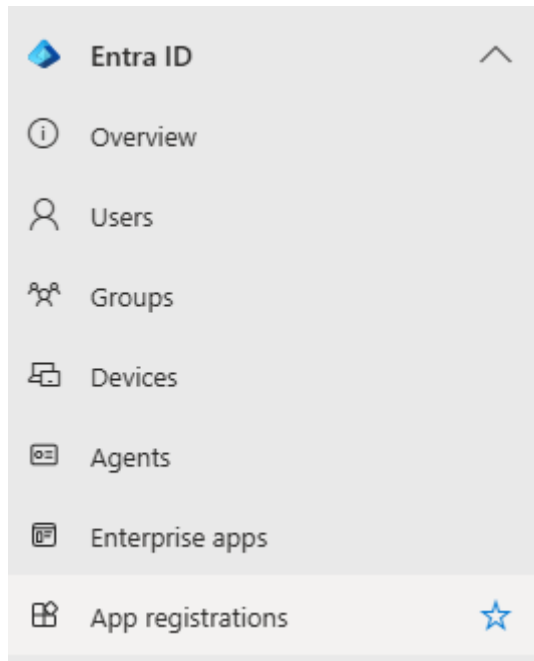
- Client Application
- Identity provider
- Callback
- Authorization code
- Tokens



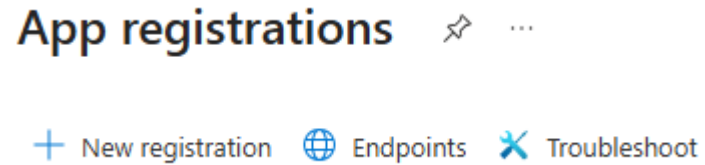
OAuth/OIDC

EXERCISE 4 – REGISTER AN APPLICATION IN ENTRA ID

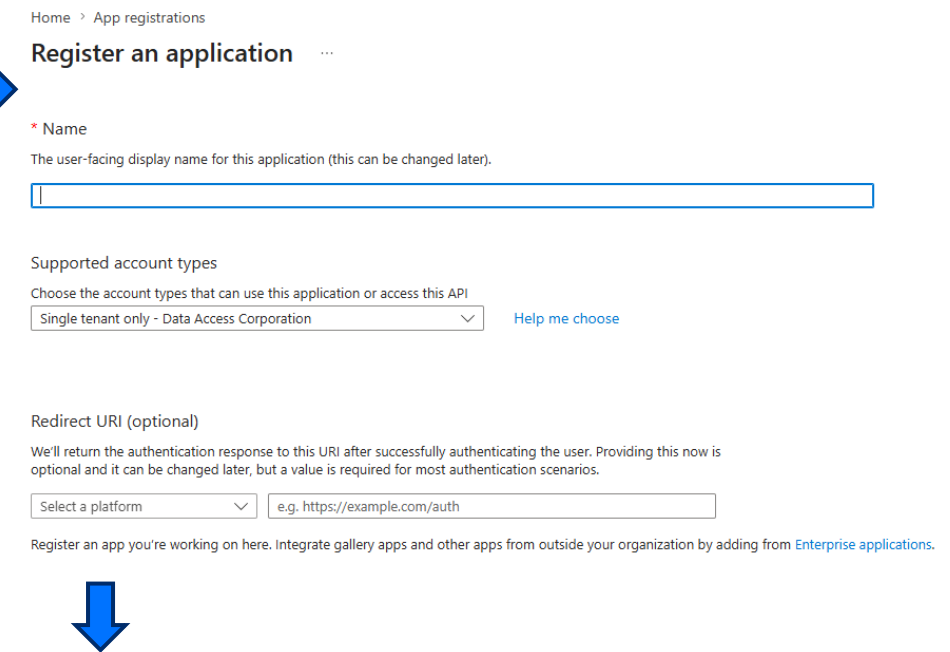
- App Registration
- Redirect URI



A screenshot of the Entra ID navigation pane. The 'App registrations' item at the bottom is highlighted with a blue star icon. A blue arrow points from the 'App Registration' bullet point in the list above to this item.

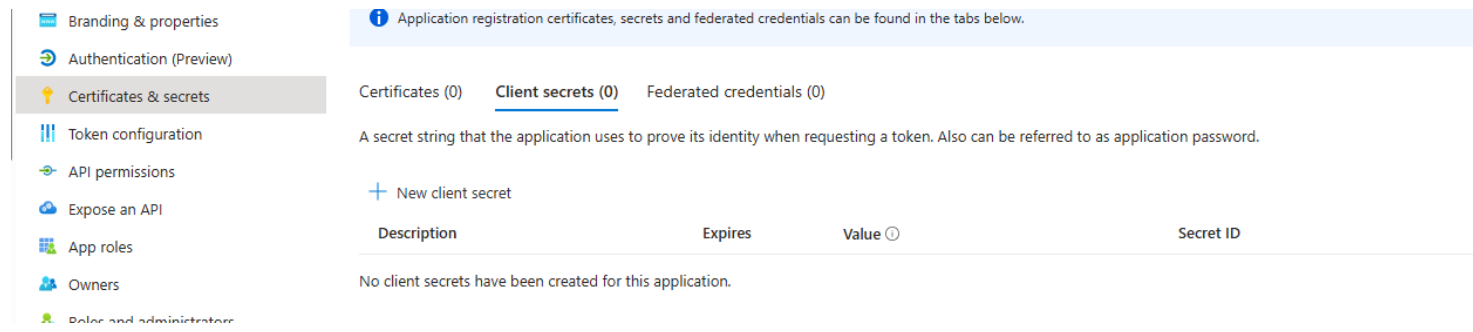


The header of the 'App registrations' page, showing the title 'App registrations' with a star icon and a menu icon. Below the title are three tabs: '+ New registration', 'Endpoints', and 'Troubleshoot'.



The 'Register an application' form in the Entra ID console. It includes a breadcrumb 'Home > App registrations', the title 'Register an application', and a form with the following fields:

- * Name**: A text input field with a placeholder for the user-facing display name.
- Supported account types**: A dropdown menu currently set to 'Single tenant only - Data Access Corporation' with a 'Help me choose' link.
- Redirect URI (optional)**: A section with a 'Select a platform' dropdown and a text input field containing 'e.g. https://example.com/auth'.



The 'Client secrets' tab in the Entra ID console. It shows a blue information banner at the top stating: 'Application registration certificates, secrets and federated credentials can be found in the tabs below.' Below this are three tabs: 'Certificates (0)', 'Client secrets (0)', and 'Federated credentials (0)'. The 'Client secrets (0)' tab is active. A description reads: 'A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.' There is a '+ New client secret' button. Below is a table with columns: 'Description', 'Expires', 'Value', and 'Secret ID'. The table is currently empty, with the text 'No client secrets have been created for this application.' below it.

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 UriParameter 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
```

```
▼ cDemoStateObject
  // Construct_Object
  // SsoLoginSuccess
  // SsoLoginFailed
  // OnSsoLoadState
  // OnSsoSaveState
  // End_Construct_Object
> cDemoStateForm
  ▼ oDemoView
    oSsoState_DD
    oStateObject1
    oStateObject2
    oWebMainPanel
    // LogOutForStateObject
    // OnBeforeShow
    // LoadStates
```

DATAFLEX SSO LIBRARY PROVIDER SETUP

- Client ID
- Client Secret
- Discovery document

DataFlex Example | Certificates & secrets

Search << Got feedback?

Overview

Quickstart

Integration assistant

Diagnose and solve problems

Credentials enable confidential applications to identify themselves to the authentication service via a certificate (instead of a client secret).

DataFlex Example

Search

Overview

Quickstart

Integration

Diagnose and solve problems

Manage

Branding & properties

Support + Troubleshooting

New support request

```
095 Object oEntraTestProvider is a cSsoProviderOpenId
096     Set psAlgorithm 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
```

```
login.microsoftonline.com/30626ff8-9106-4545-9fd8-73794e98f118/.well-known/openid-configuration
Pretty print
{
  "token_endpoint": "https://login.microsoftonline.com/30626ff8-9106-4545-9fd8-73794e98f118/oauth2/token",
  "token_endpoint_auth_methods_supported": [
    "client_secret_post",
    "private_key_jwt",
    "client_secret_basic"
  ],
  "jwks_uri": "https://login.microsoftonline.com/common/discovery/keys",
  "response_modes_supported": [
    "query",
    "fragment",
    "form_post"
  ],
  "subject_types_supported": [
    "pairwise"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256"
  ],
  "response_types_supported": [
    "code",
    "id_token",
    "code id_token",
    "token id_token",
    "token"
  ],
  "scopes_supported": [
    "openid"
  ],
  "issuer": "https://sts.windows.net/30626ff8-9106-4545-9fd8-73794e98f118/",
  "microsoft_multi_refresh_token": true,
  "authorization_endpoint": "https://login.microsoftonline.com/30626ff8-9106-4545-9fd8-73794e98f118/oauth2/authorize",
  "device_authorization_endpoint": "https://login.microsoftonline.com/30626ff8-9106-4545-9fd8-73794e98f118/oauth2/devicecode",
  "end_session_endpoint": "https://login.microsoftonline.com/30626ff8-9106-4545-9fd8-73794e98f118/oauth2/logout",
  "check_session_iframe": "https://login.microsoftonline.com/30626ff8-9106-4545-9fd8-73794e98f118/oauth2/checksession",
  "userinfo_endpoint": "https://login.microsoftonline.com/30626ff8-9106-4545-9fd8-73794e98f118/openid/userinfo",
  "kerberos_endpoint": "https://login.microsoftonline.com/30626ff8-9106-4545-9fd8-73794e98f118/kerberos",
  "refresh_token_endpoint": "https://login.microsoftonline.com/30626ff8-9106-4545-9fd8-73794e98f118/oauth2/refresh_token"
}
```

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 `cSsoApi` (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

EXERCISE 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

OAUTH SERVER LIB PREREQUISITES

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

```
004 Object o0idcHandler is a c0idcHandler
005 Procedure Setup
006   Handle hoLoginView hoConsentView
007
008   Set pho0idcProvider to o0idcProvider
009
010   Get GetLoginView of ghoWebApp to hoLoginView
011   If (hoLoginView <> C_WebUnresolvedObject) Begin
012     Set phoLoginView to hoLoginView
013     Set pho0idcProvider of hoLoginView to o0idcProvider
014     Set 01 Use 0idc\c0idcWellKnownHandler.pkg
015   End
016   Else Begin 03 Object o0idcWellKnown is a c0idcWellKnownHandler
017     Error 04 Set pho0idcProvider to o0idcProvider
018   End
019     05 Set pho0idcKeyStore to o0idcKeyStore
020     06 Set pho0idcHandler to o0idcHandler
021   Get pho0 07 End_Object
022   If (hoConsentView <> C_WebUnresolvedObject) begin
023     Set pho0idcConsentView to hoConsentView
024     Set pho0idcProvider of hoConsentView to o0idcProvider
025     Set pho0idcHandler of hoConsentView to Self
026     Send RegisterNavigateForwardPath ntNavigateBegin hoConsentView ghoWebApp ""
027   End
028   Else Begin
029     Error DFERR_PROGRAM "No consent view setup for OAuth"
030   End
End_Procedure
```

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!