# Content

## Info que nos pasan desde Solution

[‎25/‎07/‎2019 11:31] Victor Manuel Romero Rivas:

Nosotros en el startup, lo que le tenemos puesto es:  
.AddOpenIdConnect(options =>   
 options.MetadataAddress = "https://adfsdev40.everis.com/adfs/.well-known/openid-configuration";   
 options.ClientId = "el-que-te-generen"  
options.ResponseType = OpenIdConnectResponseType.Code;   
options.SaveTokens = true;   
   options.Events = new OpenIdConnectEvents                {

             OnTokenValidated = async ctx =>   
var employeeNumber =

                            ctx.SecurityToken.Claims.FirstOrDefault(claim => claim.Type.Equals("Employee-Number")).Value;  
 var applicationId =

                            ctx.SecurityToken.Claims.FirstOrDefault(claim => claim.Type.Equals("ApplicationId")).Value;   
if (string.IsNullOrEmpty(employeeNumber) || string.IsNullOrEmpty(applicationId))

                        {

                            //Escribir traza en BBDD indicando que no se pudo obtener los roles

                            //xq no están configuradas las claims necesarias en adfs 4

                        }

                        else

                        {

                            var appIdentity = GetAppIdentity(Convert.ToInt32(employeeNumber), Convert.ToInt32(applicationId));

                            if (appIdentity != null)

                            {

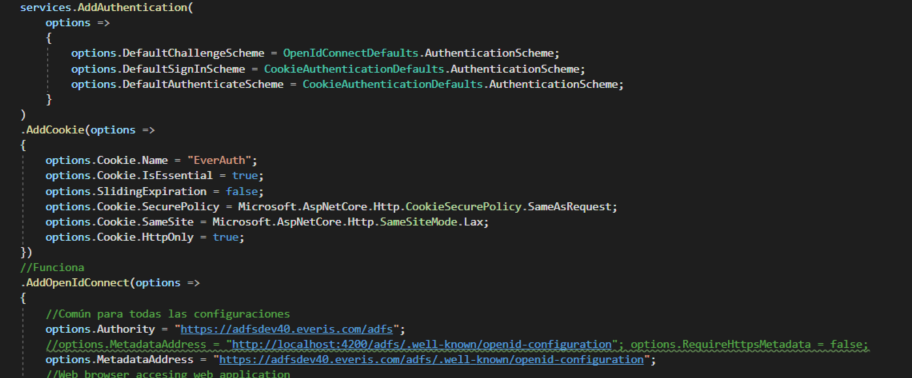
                                ctx.Principal.AddIdentity(appIdentity);

                            }

                        }

                    }

                };



**IP\_SERVIDOR:**

Desa: https://adfsdev40.everis.com/adfs/.well-known/openid-configuration   
Prod: https://adfsprod40.everis.com/adfs/.well-known/openid-configuration

Id\_client: 185dd598-f4bc-4847-9270-0f6b45db5d94 (debería de tener tb localhost:4200 pero yo no lo he probado aún)

Id\_client: f15e417a-6741-4b05-874b-92677e279f17 http://localhost:4200

## Comentarios Beltrán:

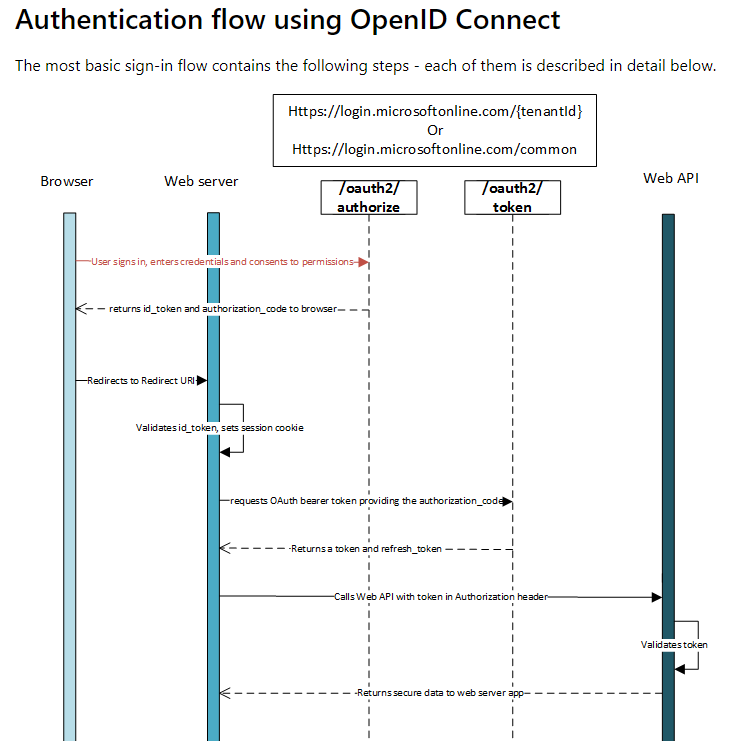
ADFS is just a Secure Token Service

You need separate instances of ADFS (auth.) and AD (user). Azure AD combines both.

ADFS is a federation layer that sits on top of AD

ADFS has the power of claims rules

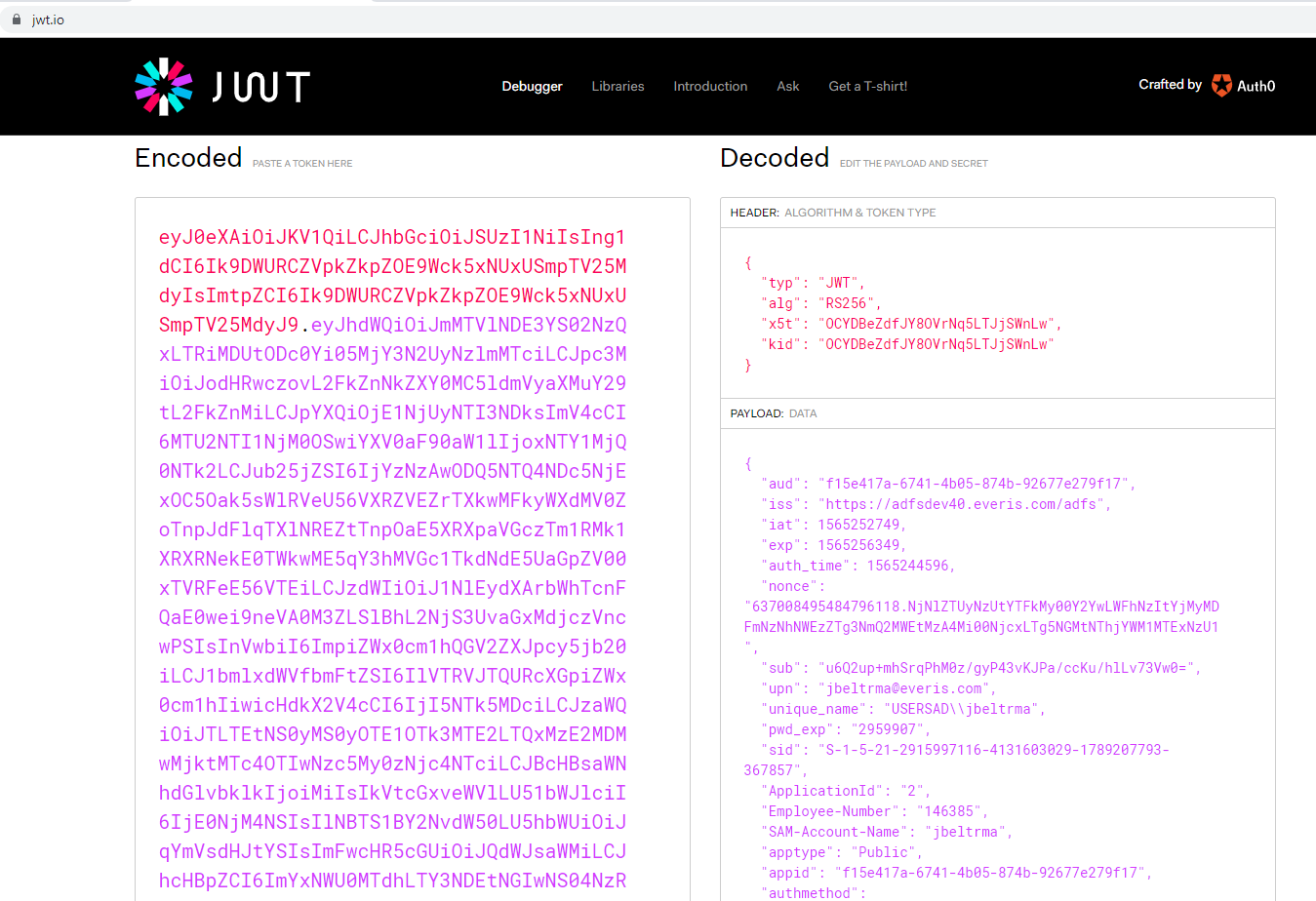
<https://docs.microsoft.com/es-es/azure/active-directory/develop/v1-protocols-openid-connect-code>



Analizando el código de la aplicación paralela:

Al realizar el login en la parte superior derecha, en el AuthorizationCodeReceiven en la variable n.JwtSecurityToken.RawData tenemos el Access\_token requerido.

Una forma de probarlo sería llevarnos dicho token a la pagina jwt.io y ver su contenido:



## Más código de referencia:

Se puede meter el access token en otro claim para mantenerlo en el server:

AuthorizationCodeReceived = async context =>

{

var tokenClient = new IdentityModel.Client.TokenClient(issuer + IdentityValues.API\_TOKEN\_URI, clientId, clientSecret);

var tokenResponse = await tokenClient.RequestAuthorizationCodeAsync(context.ProtocolMessage.Code, redirectUri);

if (tokenResponse.IsError)

{

return;

}

var userInfoClient = new IdentityModel.Client.UserInfoClient(issuer + IdentityValues.API\_USER\_INFO\_URI);

var userInfoResponse = await userInfoClient.GetAsync(tokenResponse.AccessToken);

var identity = new ClaimsIdentity();

identity.AddClaims(userInfoResponse.Claims);

identity.AddClaim(new Claim(IdentityValues.CLAIM\_TYPE\_TOKEN\_ID, tokenResponse.IdentityToken));

identity.AddClaim(new Claim(IdentityValues.CLAIM\_TYPE\_TOKEN\_ACCESS, tokenResponse.AccessToken));

if (!string.IsNullOrEmpty(tokenResponse.RefreshToken))

{

identity.AddClaim(new Claim(IdentityValues.CLAIM\_TYPE\_TOKEN\_REFRESH, tokenResponse.RefreshToken));

}

var nameClaim = new Claim(ClaimTypes.Name, userInfoResponse.Claims.FirstOrDefault(

c => c.Type.ToLower() == IdentityValues.SAM\_ACCOUNT\_NAME.ToLower())?.Value);

identity.AddClaim(nameClaim);

context.AuthenticationTicket = new AuthenticationTicket(

new ClaimsIdentity(identity.Claims, context.AuthenticationTicket.Identity.AuthenticationType),

context.AuthenticationTicket.Properties);

}

Otra posible implementación para meterlo en una claim:

<https://developer.okta.com/blog/2018/08/29/secure-webforms-with-openidconnect-okta>

También podemos meterlo en una cookie de la forma habitual.

* Hence = por lo tanto

We wanted the same for other assets, like the SAML token handlers, hence we created a very thin layer

* Despite all of those advancements= a pesar de todo esos avances

Despite all of those advancements, however, the way in which web application developers interact with claims based identity has remained the same for all this time