Fwk Service Dispatcher Web API & Authentication

En el presente documento se detalla la implementacion los cmponentes intervinentes en el workflow de autenticacion.

La autenticación esta basada en JWT <https://jwt.io/introduction/> donde se define una forma compacta autocontenida para transmitir la informacion entre clienet y servidor (recurso y/o de seguridad). Esta informacion esta digitalmente firmada utilizando HMAC , RSA o **ECDSA**



**Resource Owner:**

The entity or person (user) that owns the protected **Resource**.

**Client Applications:**

Es cualquier app que (software) solicite acceso a algun recurso protegido en el Resource Server (auidience)

Ver Registro de Client Applications en Resource Owners

**Resource Server (Audience):**

Es el servidor que contiene las api que permiten el acceso directo a los recursos portegidos.

Nombre fisico del fuente fwk**Dispatcher**WebApi

**Authorization Server (Issuer)**

Es el server que genera y verifica las credenciales del usuario, genera el token y lo retorna al cliente en caso de éxito. Para ello requiere chequar el registro del Resourse Owner

Nombre físico del fuente fwk**Authentication**WebApi

**APIS que provee** **Authorization Server (Issuer)**

### 1 Grant Tocken - Conceder un JWT

POST <https://localhost:51000/oauth/token> Este metodo es el principal y es el que genera el token

Host: localhost: 51000

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

**Parametros**:

* **username** . nombre de usuario con el que esta registado en la bd de negocio
* **password**: password del usuario
* **grant\_type** : password
* **client\_id:** Nombre o identificador de la aplicacion en cuestion que utilizara el Issuer para firmar el jwt: Recordar que el Issuer puede ser generador de tokens para mas de una app.
* **client\_secret**: clave secreta con la que se gfenero el Hash que

Nota: Para que un client\_id sea valido este debera ser previamente registrado y accesible al **Authorization Server**

### 2 Registro de Client Applications en Resource Owners

Las apps clientes se almacenan en una tabla propuesta por el bloque de seguridad Fwk.Security.Identity llamada SecurityClients (en otro apartado se muestra como instalar las tablas de este bloque)

Para efectuar el registro se debe llamar al siguiente Endpoint:

POST /api/account/createApplication HTTP/1.1

Host: localhost: 51000

Content-Type: application/json

{

"**Id**":"webChatCustomer", //Identificador de la aplicacion

"**Secret**":"socratesSecret", //

"**Name**":"CustommersChat", //Nombre descriptivo de la aplicacion

"ApplicationType":"1",

"Active":true,

"RefreshTokenLifeTime":20, //tiempo de expiracion del token jwt generado

"AllowedOrigin":1

}

Por lo tanto llamando a esta api se permite crear una aplicacion en SecurityClients Donde SecurityClients es un repositorio accesible a travez del bloque Fwk.Security.Identity para registrar todas las App que requieran autenticacion y autorizacion de usuarios

### 3 Refresh Token

[lo describiremos en porximas versiones del doc]

POST /oauth/token HTTP/1.1

Host: localhost:51028

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

cache-control: no-cache

Postman-Token: 88f53160-ec04-4c67-bdd9-843cea697199

**Parametros**

* **grant\_type** : refresh\_token
* **client\_id**: Nombre o identificador de la aplicacion en cuestion que utilizara el Issuer para firmar el jwt: Recordar que el Issuer puede ser generador de tokens para mas de una app.
* **refresh\_token**: Este valor es un token con menor tiempo de expiraciopn que el access\_token explicado anteriormente y es utilizado para refrescar automaticamente en el servidor de autenticacion el token Ej 2ba48dc5530946cba4cf6d778a46ceb0
* **client\_secret**: clave secreta con la que se gfenero el Hash que

**APIS que porvee Resource Server (Audience)**

El dispatcher porvee la capacidad de ejecutrar un servicio de la arquitrectura FWK y para ello expone un API unica de ejecucion

POST /api/fwk/execute/ HTTP/1.1

Host: localhost:52000

**Content-Type**: application/json

**Authorization header Token** : Aqui enviamos el acces\_tocken generado al llamar a la API del

**Body** : El body del api Execute es un objeto que contiene toda la informacion necesaria para ejecutar el servicio de la capa SVC en el orquestador o dispatcher

Ejemplo:

{

serviceProviderName : ‘HealthMetadataProvider’

serviceName: ’RetrivePatientsService’

jsonRequest: ‘serializacion json de un objeto Requets de la arquitectura FWK’

}

***Ejemplo body para la ejecución de un servicio***

**serviceProviderName** Nombre del proveedor de servicio FwkMetadatta Provider configurado en el Dispatcher (Orquestador-host y en este caso el Resource Server) : Es basicamente el porveedor que tiene el conocimiento de la Metadata se servicios SVC

**jsonRequest** Cabe aclarar aqui que es importante haber entendido la arquitectura de como se conforma un servicio Requets y Reponse

En este caso se pide ejecutar el servicio SVC *RetrivePatientsService* donde se pasan una BE que funcionaria como filtro en el servicio : El filtro retorna todos los usuarios que contengan nombre:marcelo apellido oviedo sin importar dni y retorna tambeijn aquellos en estado desactivado.-

El **ContextInformation** es un objeto que siempre viaja en el requts y no debe ser construido por un porgramador final sino mas bien es responsabilidad del Wrapper que se comunica con la API der Resource Server

{

"**serviceProviderName**": "",

"**serviceName**": "RetrivePatientsService",

"**jsonRequest**": {

"**SecurityProviderName**": null,

"CacheSettings": {

"CacheOnServerSide": false,

"CacheOnClientSide": false,

"ResponseCacheId": null,

"TimeMeasures": 0,

"ExpirationTime": 0,

"CacheManagerName": null

},

"**ServiceName**": "RetrivePatientsService",

"BusinessData": {

"Nombre": "marcelo",

"Apellido": "oviedo",

"NroDocumento": null,

"IncludeInactive": true

},

"**ContextInformation**": {

"Culture": "es",

"ProviderNameWithCultureInfo": null,

"HostName": null,

"HostIp": null,

"HostTime": "\/Date(-62135586000000)\/",

"ServerName": null,

"ServerTime": "\/Date(-62135586000000)\/",

"UserName": null,

"UserId": null,

"AppId": "health",

"ProviderName": null

}

}

}

**Configurar el** **Authorization Server (Issuer)**

**Archivo de configuracion**

**appSettings.secConf**ig = “ruta donde se encuentra el archivo que contiene los porveedores de autenticacion”

<add key="secConfig" value="c:\projects\health\securityConfig.json" />

En otro apartado se explicara en que consiste un Security JWT Provider

**connectionStrings**

Configurar tantas cadenas de conección como sean necesarias : Estas cadenas son referenciadas por los componentes de autenticacion : en este caso de ejemplo se trata de un ***SecurityManager*** que proviene del bloque Fwk.Security.Identity y utiliza una estruyctura similar (modificada) de Identity ASP Net 2.0.

Las cadenas de conexion son referenciadas desde los diferentes providers de seguridad bajo el atributo "securityModelContext": "health"

**Proveedor de seguridad**

Se configura en el archivo securityConfig.json

En el siguiente json se muestra un conjunto de porveedores a los que el AuthorizationServer (issuer) da soporte

{

"providers": [

{

"name": "healthTesting",

"description": "Proveedor health wcf",

"audienceId": "pelsoft",

"issuer": "https://200.30.1.1:51000",

"audienceSecret": "qMCdFDQuF23RV1Y-1Gq9L3cF3VmuFwVbam4fMTdAfpo",

"securityModelContext": "health"

},

{

"name": "healthProd",

"description": "Proveedor health w32",

"audienceId": "pelsoft",

"issuer": "https://200.30.1.1:51000",

"audienceSecret": "qMCdFDQuF23RV1Y-1Gq9L3cF3VmuFwVba m4fMTdAfpo",

"securityModelContext": "healthProd"

}

]

}

Notemos aquí que el Authorization Server (issuer) es el mismo para todos y es un dato muy importante para ser validado despues por el Resourse server cuando consuma el JWT.-

**securityModelContext**: Este atributo apunta a un contexto de seguridad y para el caso de SecurityManager es la cadena de conexion donde estan las tablas de usuarios roles categrias y registros de clientes que sconsumen jwt