Document d'adhesions a la Plataforma de Seguretat (PdS)

Historial Electrònic de Salut (HES)

Versió 1.1.3
Gener 2025

Generalitat de Catalunya
Departament de Salut

Sistema de Salut de Catalunya

Revisió	Redactat per	Revisat per	Aprovat per	Data aprovació	Data publicació
1.0.3	Miguel Medina	Joan Esteve	Joan Esteve	22/02/2023	22/02/2023
1.0.4	Modesto Román, Adrián Toca Raúl Porcar				
1.0.6	Raúl Carretero				
1.0.7	Raúl Carretero	Joan Esteve	Joan Esteve	25/03/2024	25/03/2024
1.1.0	Joan Esteve	"	"		
1.1.1	Raúl Carretero				
1.1.2	Raúl Porcar				26/11/24
1.1.3	Joan Esteve	"	"	17/01/2025	17/01/2025

Registre de Canvis

Revisió	Apartat	Data Modificació	Motiu del canvi
1.0.3	1, 2, 3, 4	22/02/2023	Versió inicial
1.0.4	5.1.3, 5.1.4, 5.1.5	17/07/2023	Afegim apartat d'alta, llistat i esborrat d'usuaris a través d'API. Canviem codi projecte per 3982.03.
1.0.5	1.2	21/09/2023	S'afegeix el punt 1.2 Com adherir un nou sistema de Salut a la PDS
1.0.6	5.2, 5.3. 5.4	09/01/2024	El punt 5.2 actual el desplacem al 5.4. Afegim en l'apartat de com configurar un client SAML (5.2) Afegim l'apartat d'Integració amb GICAR (5.3)
1.0.7	4.1	25/03/2024	Indiquem en el punt 4.1 que necessitem per poder fer una pantalla de login (o altres pantalles) per als clients amb flux ACF
1.1.0	Tots	01/09/2024	Revisió general del document Explicació del login amb credencials de GICAR o VALID Explicació obtenció tiquets SAML per OSB de Salut Explicació flux de logout
1.1.1	2.1.5 4.4 5.1.2	28/10/2024	Explicació de la nova funcionalitat amb la petició UserInfo a la PDS
1.1.2	1.2	26/11/2024	Canviem correu Oficina d'adhesions
1.1.3	2.1 4.1	17/01/2025	Informació del flux ACF amb PKCE Explicació del flux de logout sense id_token_hint

RESPONSABLE DEL DOCUMENT: AM13 / Joan Esteve

ÍNDEX

1. Introducció 4
1.1 Objecte del document 4
1.2 Com adherir una nova iniciativa de Salut a la PDS 4
1.2.1 Petició de Realm o client de tipus OpenID Connect 5
1.2.2 Petició de Realm de tipus SAML2 6
2. Procés d'autenticació i autorització mitjançant PDS 8
2.1 Tipus de fluxos d'autenticació suportats 8
2.1.1 Authentication Code Flow (ACF) 9
2.1.2 Client Credentials Flow (CCF) 9
2.1.3 Refresh Token Flow 10
2.1.4 Logout Flow OIDC 10
2.1.5 User Info Endpoint 10
2.2 Altres configuracions en el Realm i client 11
2.3 Integració amb sistemes d'identitat corporatius 11
2.3.1 Delegació del login a GICAR (directori de professionals) 11
2.4 Delegació del login de VALID (ciutadans) 13
2.5 Validació del token d'accés i autorització en els backends destí 14
3. Accions requerides a PDS per admetre l'adhesió (proveïdor) 16
4. Detall de peticions de token d'accés segons el tipus de flux OIDC 17
4.1 Login amb client ACF 17
4.1.1 Personalització dels estils de la pantalla de login 17
4.2 Generar token d'accés amb client CCF 19
4.3 Petició Refresh per obtenir un nou token 19
4.4 Petició User Info per obtenir els atributs de la sessió 21
4.5 Exemple de petició de logout 21
4.6 Paràmetres addicionals a la petició per generar el token. 22
4.6.1 Exemple de paràmetres sense validació 22
4.6.2 Paràmetres que es validen contra llistes tancades locals 23
4.6.3 Paràmetres que es validen contra catàlegs del Servidor Terminològic: 23
4.7 Validació del token mitjançant la clau pública. 23
5. Apèndix 25
5.1 Exemple de peticions 25
5.1.1 Petició CCF 25
5.1.2 Petició User Info 27
5.1.3 Petició ACF 28
5.1.4 Alta d'usuaris 32
5.1.5 Llistat d'usuaris 33
5.1.6 Esborrat d'usuaris 34
5.2 Petició de tiquet SAML 34
5.3 Referències 36

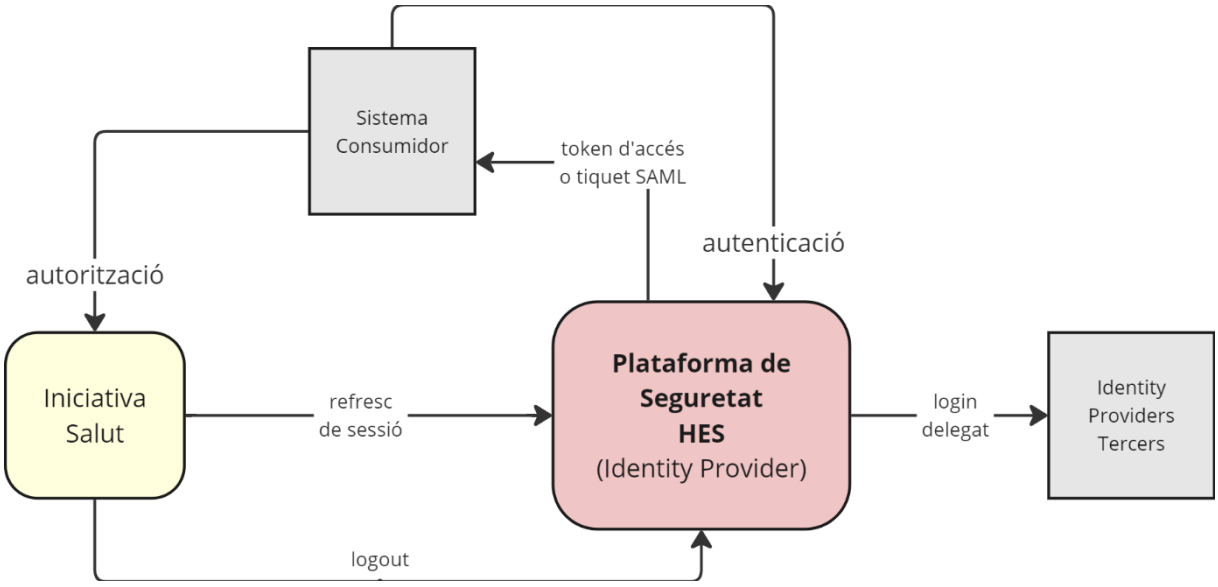
1. Introducció

1.1 Objecte del document

El present document mostra la guia per facilitar l'adhesió de noves iniciatives o sistemes de la nova Història Electrònica de Salut (HES).als serveis que ofereix la **Plataforma de Seguretat HES (PdS)** com a *Identity Provider* transversal de Salut.

Aquests serveis comprenen principalment la gestió de:

- **l'autenticació**: les credencials necessàries per consumir serveis o dades d'una iniciativa de Salut
- **l'autorització:** el control del permís sobre aquests serveis exposats per la iniciativa, segons les credencials presentades per el consumidor.



Es basa en la distribució *open source* de **Keycloak** com a gestor d'identitats i control d'accés (*Identity Provider*), i suporta els estàndards **OpenIdConnect** (OIDC) i **SAML2**.

La definició de la seguretat amb atributs **ABAC** (*Attribute Access Control*) s'integra amb IBM **Operational Decision Manager** (*ODM*) com avaluador de regles d'accés. Keycloak (PdS) envia els atributs de context de la petició a l'ODM i aquest, en funció de les regles de seguretat configurades, li retorna els atributs de seguretat HES definits.

1.2 Com adherir una nova iniciativa de Salut a la PDS

Per incorporar una nova iniciativa, solució, o domini funcional de Salut a l'autorització de la PDS, primer cal enviar una sol·licitud d'adhesió a la següent bústia:

Oficina de Solucions TIC per al SISCAT (oficina.solucionstic@catsalut.cat)

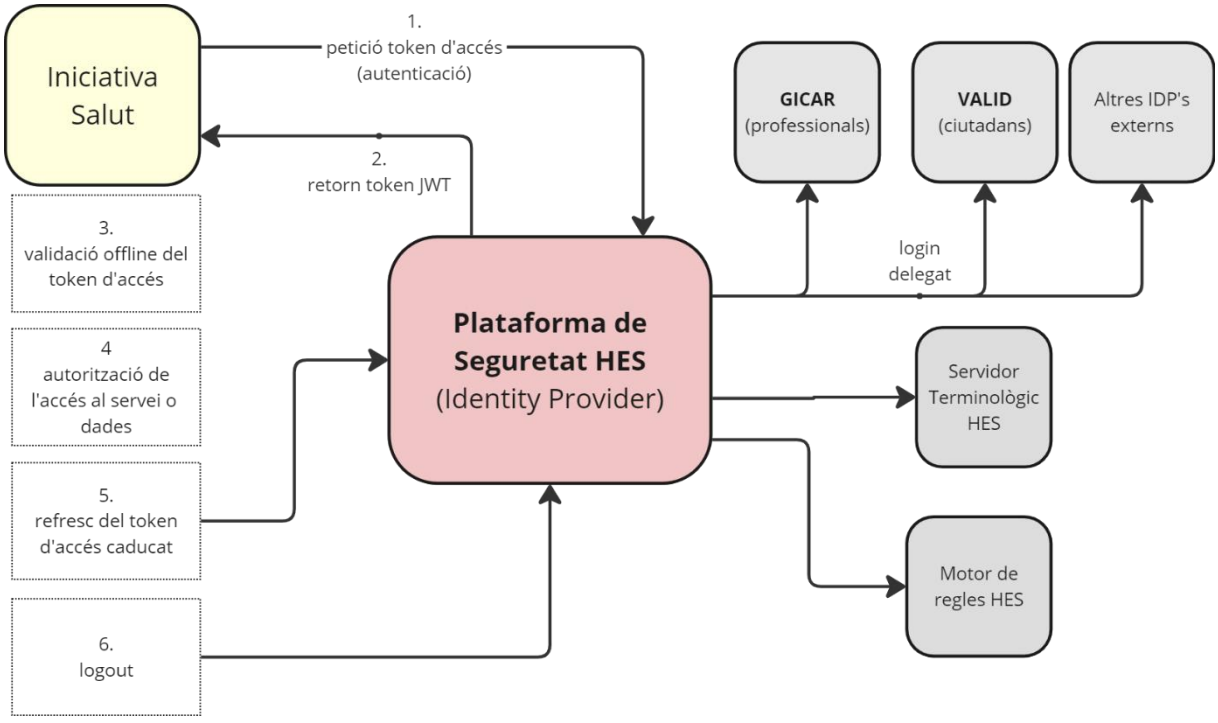
Es facilita aquest **Manual d'adhesions** a l'equip de desenvolupament de la iniciativa a adherir-se, perquè faci l'anàlisi de necessitats de la seva autenticació i autorització: petició d'un *Realm* i *Clients* dins de PDS, tipus de *Realm* a configurar, flux d'autenticació requerit en cada *Client*, també si cal fer una càrrega inicial d'usuaris, federació (delegació) del login a un *Identity Provider* tercer, o habilitar unes credencials específiques per gestionar l'alta d'usuaris a través de l'API de Keycloak.

Un *Realm* és una agrupació de *Clients* que comparteixen un endpoint únic per demanar o renovar tokens d'accés, on aquests comparteixen una mateixa clau i algorisme de signatura.

Habitualment, s'aconsella tenir un *Realm* per sistema d'informació, de manera que malgrat s'adrecin tokens per clients o microserveis diferents d'aquest sistema, la manera de validar aquests tokens sigui la mateixa, doncs estan signats amb els mateixos certificats.

1.2.1 Petició de Realm o client de tipus OpenID Connect

Aquest serà l'operació més habitual. Una iniciativa de Salut sol·licita un **Client** en un **Realm** ja existent a la PDS, o un **Realm** propi nou, per executar fluxos d'autenticació i autorització de tipus **OpenId Connect** (OIDC).



En aquest cas, l'equip de PDS facilita l'endpoint amb les operacions OIDC disponibles en el *Realm*. Es pot agafar com exemple el *Realm* "PDS-Dummy" que tenim a Preproducció:

```
https://preproduccio.pds.hes.catsalut.gencat.cat/auth/realms/PdS-DUMMY/.well-known/openid-configuration
```

Existeix també la possibilitat de configurar paràmetres addicionals en la petició de token d'accés (respectant l'estructura JWT d'aquest), que després s'incorporen com a *claims* (atributs). En aquest cas, el sol·licitant del *Client* a la PDS ha d'indicar:

- Obligatorietat dels paràmetres. Admetem cardinalitats: 0..1, 1, 0..N o 1..N.
- Possibilitat de fixar la validació dels valors permesos en cada paràmetre: contra catàlegs del Servidor Terminològic FHIR o contra una llista tancada de valors.
- Crides al motor de regles ODM per calcular *claims* addicionals segons els paràmetres d'entrada incorporats en la petició de token d'accés.

Consideracions a tenir en compte:

- No es podrà donar d'alta un paràmetre nou si ja existeix un altre prèviament creat amb un significat semàntic similar: *tipus_usuari* si ja tenim *user_type* per exemple.
- El nom de cada paràmetre serà en anglès.
- Un cop acordats, es presenta l'estructura de *claims* dins el token JWT resultant al lot aplicacions, perquè puguin planificar el desenvolupament de l'autorització dels seus serveis o microserveis.

L'equip de PDS dona d'alta el *Realm* o els *Clients* sol·licitats, i si aplica: configura la incorporació de nous paràmetres al token (amb les seves validacions), la invocació de regles al motor, o la importació d'usuaris via API de Keycloak.

Finalment, es facilita a la iniciativa adherida les URL del *Realm* a cada entorn, i la relació de **client_id / client_secret** generats. A més, es proporciona un exemple de petició POST per obtenir el token d'accés al *Client* en el *Realm* creat.

La iniciativa sol·licitant haurà de gestionar les regles de tallafocs corresponents per accedir per HTTPS als endpoints de la PDS.

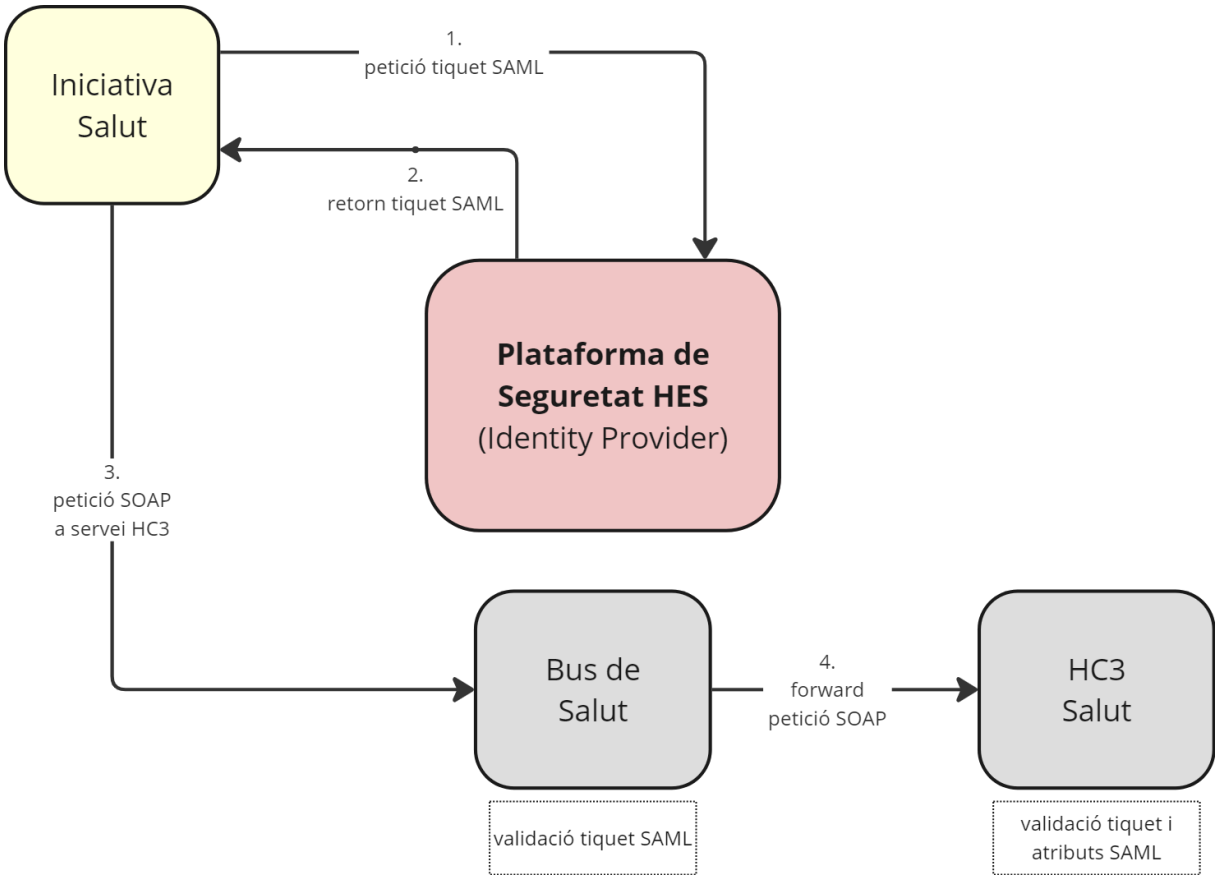
El sistema sol·licitat és responsable de custodiar les credencials facilitades per l'equip de PDS (*client_secret*). En cas que aquestes credencials siguin compromeses, cal contactar amb els responsables de PDS per gestionar la revocació d'aquestes i regeneració de noves.

1.2.2 Petició de Realm de tipus SAML2

La PDS també permet generar *Realms* per integracions basades en l'estàndard SAML2.

Actualment, la principal utilitat d'aquesta opció és que diposem d'un *Realm* que genera tiquets SAML vàlids per invocar els serveis d'HC3 a través del **Bus d'integracions de Salut**.

Aquests serveis d'HC3, que són de tipus SOAP, requereixen un *SoapENV Header* amb una capçalera *WebServiceSecurity* on cal incloure un tiquet SAML. Per tant, entre HC3 i PDS s'han configurat les claus privades de certificats per permetre demanar aquests tiquets SAML a PDS i que tant el Bus de Salut com HC3 els consideri correctes.



Això ofereix un principal avantatge a les iniciatives de Salut que necessiten invocar serveis d'HC3, i és que no cal demanar ni mantenir el seu propi certificat d'aplicació per signar el tiquet SAML, ni demanar la instal·lació al Bus i a HC3, ni implementar la creació dels tiquets SAML, doncs els poden demanar de forma segura directament a la PDS.

L'equip de la PDS proporciona unes credencials úniques a cada iniciativa dins aquest *Realm*, aquestes no es comparteixen entre sistemes d'informació diferents.

2. Procés d'autenticació i autorització mitjançant PDS

La PDS es troba disponible en dos entorns no-productius (Integració i Preproducció) i un entorn de Producció, en les següents adreces:

Entorn	URL Keycloak
INT (només amb visibilitat intranet)	https://integracio.pds.hes.catsalut.intranet.gencat.cat
PRE	https://preproduccio.pds.hes.catsalut.gencat.cat
PRO	https://pds.hes.catsalut.gencat.cat

PDS actua com a proveïdor d'autenticació (*Identity Provider*) per les iniciatives adherides, facilitant als clients els tokens d'accés segons les credencials proporcionades.

Per altra banda, cada iniciativa gestiona l'autorització dels serveis que publica, o les dades retornades en aquests, a través del contingut del token: atributs (*claims*) o permisos.

2.1 Tipus de fluxos d'autenticació suportats

En la PDS es poden configurar els següents fluxos d'autenticació:

- **Authorization Code Flow (ACF):** és el cas en que el subjecte que s'autentica és un usuari amb unes credencials pròpies, habitualment username + password.

  En aquest cas, la PDS ofereix la pantalla de login a la iniciativa, i fa una redirecció acordada a un endpoint destí en cas que les credencials siguin correctes.

  D'aquesta manera, la iniciativa no requereix tenir visibilitat d'aquestes credencials que l'usuari introdueix. També es pot delegar el login a sistemes com GICAR o VALID, tal com explicarem a continuació.

  La pantalla de login de la PDS es pot customitzar a nivell d'estils, imatges, icones o textos, en base a generar un "tema" de frontend especial per aquest client. En altre cas, sortiran els estils per defecte de Keycloak.

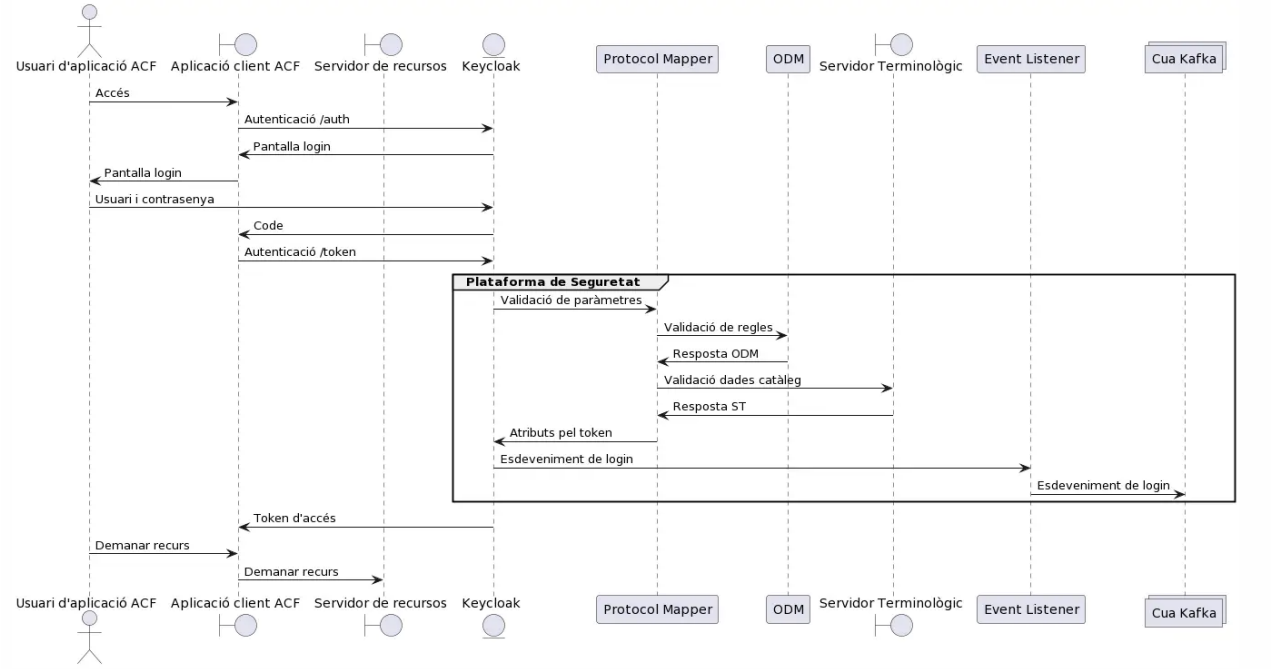
  Des de la PDS suportem també incorporar PKCE que és una millora sobre ACF a l'hora de prevenir atacs CSRF o injecció de resposta de tokens no autoritzada.

- **Client Credentials Flow (CCF):** és el cas en que el subjecte que s'autentica és un sistema. En aquest escenari, les credencials que dona la PDS són fixes: un **client_id** i un **client_secret** que la iniciativa ha de custodiar per demanar els tokens d'accés.

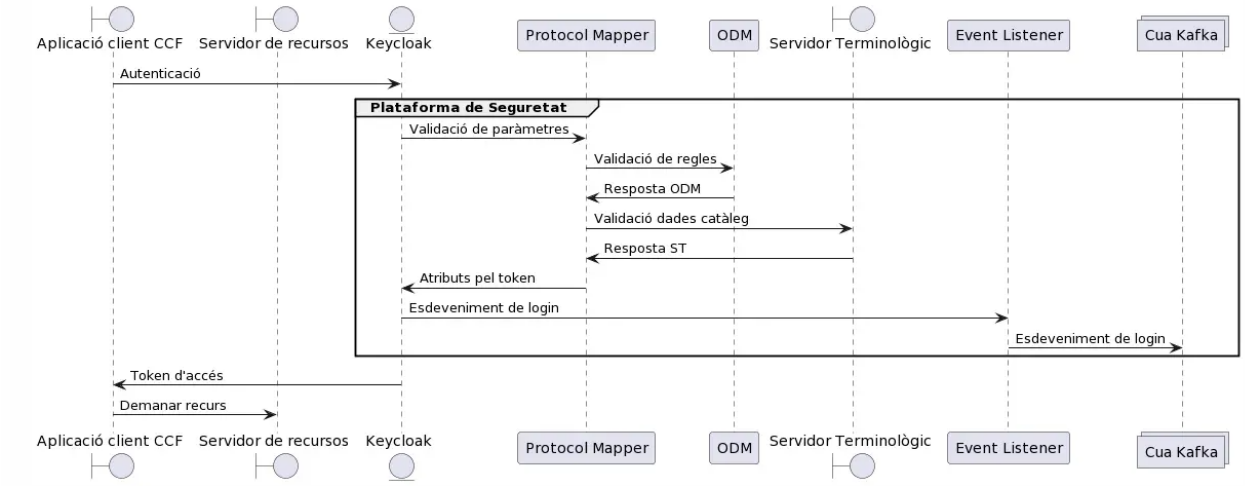
  Aquesta petició també pot venir acompanyada de paràmetres addicionals (*claims*) o validació de motor de regles.

- **Refresh Token Flow:** en el cas en que un token d'accés ha caducat, l'estándar OIDC ofereix també el token de refresc (*refresh_token*) per generar un token d'accés nou de manera que l'usuari no tingui que autenticar-se de nou.

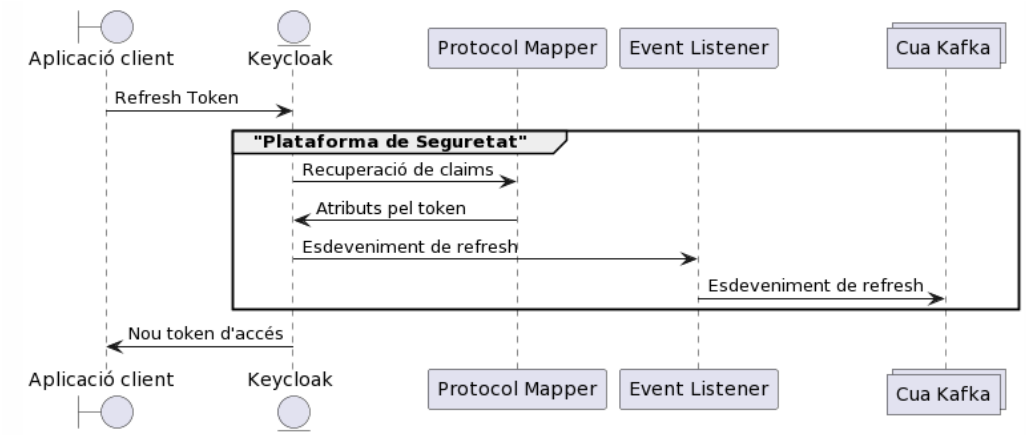
2.1.1 Authentication Code Flow (ACF)



2.1.2 Client Credentials Flow (CCF)



2.1.3 Refresh Token Flow



2.1.4 Logout Flow OIDC

El Realm de la PDS exposa també l'endpoint per realitzar el logout (finalitzar la sessió, i per tant *els* tokens d'accés o de refresc a aquesta sessió queden sense utilitat)

És important tenir en compte que el flux de Logout és una implementació específica d'OIDC; per tant, requereix que la petició de token d'accés inclogui l'*scope* = **openid**

Això generarà tres tokens que la iniciativa haurà de mantenir:

- **access_token**: el token necessari per invocar els serveis en la iniciativa destí
- **refresh_token**: el token necessari per generar un nou *access_token* en cas que aquest hagi caducat (per defecte 15 minuts)
- **id_token**: un token amb menys informació, simplement identifica la sessió dins de Keycloak per poder executar el flux de logout.

La petició de logout haurà d'incloure aquest darrer *id_token*. La PDS, un cop alliberada la sessió, executarà una redirecció GET a la URL indicada.

En cas de tenir integració amb el login de tercers, és possible que aquesta URL sigui pròpia del destí, en cas de requerir alliberar altres dades de sessió o cookies relacionades.

També existeix l'opció de NO enviar el id_token en la petició de login. En aquest cas, el logout no serà directe, i el Keycloak de la PDS oferirà un frontend que demanarà al usuari confirmar l'operació de logout (ja que necessita llegir la sessió de la *cookie* de l'usuari).

2.1.5 User Info Endpoint

Cada *Realm* també disposa d'un endpoint OIDC de tipus "*User Info*", que permet recuperar els atributs que consten en la sessió de PDS en base a un token d'accés vigent.

Es pot configurar per a reduir la quantitat d'atributs o *claims* que viatgen en el token d'accés JWT (informació que segurament no és rellevant per realitzar autorització en els backends de

les iniciatives, com per exemple Noms i Cognoms...) i recuperar aquesta informació via endpoint de User info.

2.2 Altres configuracions en el Realm i client

L'equip de PDS realitza les següents configuracions opcionals en el *Realm* o *Client*, si ecau:

- **Configuració del temps d'expiració** del token d'accés i del token de refresc. Per defecte, tenen una caducitat de 15 i 30 minuts respectivament
- **Configuració del Protocol Mapper**, per la validació de paràmetres addicionals en la petició de token d'accés (contra el Servidor Terminològic HES o contra llistes tancades de valors) i la invocació de regles al motor de regles, per incorporar aquesta informació com a *claims* en el token JWT.
- **Configuració de l'Event Listener**, per tal de generar de forma asíncrona una auditoria de les peticions de token d'accés a la Plataforma d'Auditoria HES.
- **Configuració de Keycloak Metrics**, per generar mètriques necessàries per la monitorització de l'activitat i l'observabilitat de la plataforma PDS.

2.3 Integració amb sistemes d'identitat corporatius

La PDS permet implementar el flux d'autenticació d'usuaris delegat a dos sistemes d'identitat corporatius de la Generalitat de Catalunya:

- **GICAR**: adreçat al directori de professionals
- **VALID**: adreçat al directori de ciutadans

Aquesta integració pot funcionar, de fet, en una modalitat híbrida: permetent a la iniciativa adherida a la PDS treballar amb GICAR o VALID, o tots dos al mateix temps.

2.3.1 Delegació del login a GICAR (directori de professionals)

GICAR és l'eina amb què s'ha dotat a la Generalitat de Catalunya per a la gestió centralitzada de les identitats de persones que interactuen amb els sistemes d'informació.

Facilita la gestió de les identitats (persones) que treballen o col·laboren amb la Generalitat de Catalunya (funcionaris, interins, empreses públiques, externs, etc.), així com la gestió i el control de l'accés als recursos per part de les identitats -entenent com a recurs qualsevol eina o sistema d'informació necessari perquè el professional desenvolupi les seves funcions.

En aquest context, la PDS ofereix un *Realm* específicament dedicat per a que els usuaris s'autentiquin amb les seves credencials de GICAR, mitjançant un flux de tipus ACF (*Authorization Code Flow*) amb un client que s'hi hagi incorporat.:

En aquest escenari, quan un usuari sol·licita una petició d'autenticació, la PDS fa automàticament una redirecció a GICAR, presentant la seva pantalla de login:

És important destacar que GICAR és un servei ofert per Identitat Corporativa CTTI, i nosaltres des de PDS només consumim la seva passarel·la d'autenticació delegada, però no som responsables del *Identity Provider* destí.

En cas d'autenticar-se correctament l'usuari, s'executa una redirecció GET a la *redirect_uri* pactada entre la iniciativa i PDS, per prosseguir amb el flux per obtenir el token d'accés.

La PDS recupera els següents paràmetres del login delegat a GICAR:

- Codi d'usuari: en aquest cas és el DNI
- Nom i cognoms
- Correu electrònic
- GroupMembership: llista de grups GICAR associats a l'usuari autenticat
- Unitat Major / Unitat Menor: són atributs del directori corporatiu de GICAR

També es poden recuperar altres atributs que l'usuari tingui informats GICAR, sempre que s'hagin gestionat aquests des del sistema **Control d'Accés de Recursos** (CAR) de GICAR. Aquest sistema és una eina web que permet definir i assignar atributs (recursos) a usuaris, i que un usuari amb privilegis d'administrador de CAR de GICAR pot gestionar.

https://gicar.intranet.gencat.cat/gdi/controlaccesrecursos/

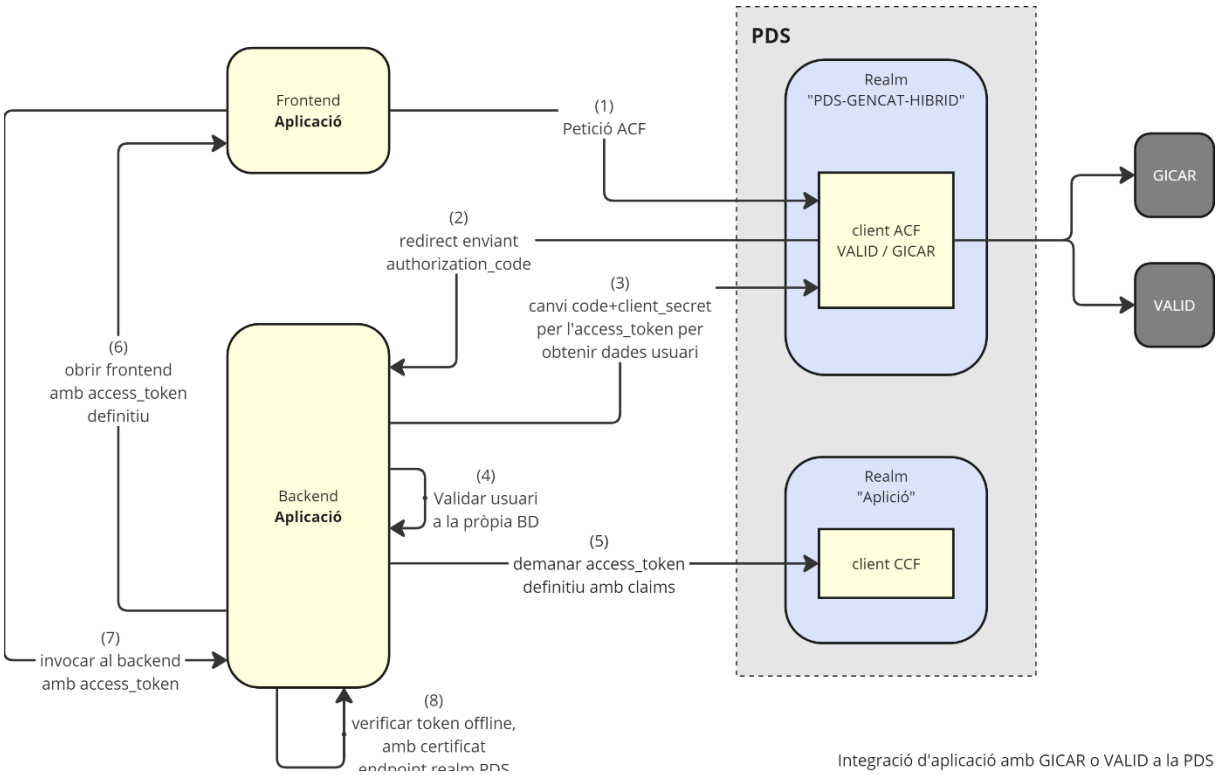
Tots aquests paràmetres recuperats es poden incloure com a *claims* en el token d'accés.

Per altre banda: NO podem incorporar atributs o *claims* addicionals directament enviats en la petició. Si la iniciativa adherida requereix fer algun control de l'autorització basada en altres atributs de l'usuari, ho ha de fer des del seu propi backend.

Per aquest motiu, en cas que la iniciativa tingui la necessitat d'incloure més contingut en el seu token, habitualment aprovisionem un *Realm* específic de la iniciativa, i aquesta bescanvia

el primer token d'accés del flux ACF original per un nou token d'accés en un flux CCF on ja pot personalitzar tots els paràmetres o execució de regles necessàries.

El següent diagrama descriu aquesta interacció:



Integració d'aplicació amb GICAR o VALID a la PDS

2.4 Delegació del login de VALID (ciutadans)

VALID és l'integrador de serveis d'identitat digital de Catalunya perquè la ciutadania tingui totes les opcions per tramitar fàcilment.

La PDS ofereix, en el mateix *Realm* indicat en l'apartat anterior, la possibilitat d'incorporar clients on es delegui el login a VALID, mitjançant també un flux ACF.

En aquest cas, la PDS ofereix la pantalla de login de VALID, presentant diferents opcions d'autenticació a l'usuari com per exemple IDCAT Mòbil o DNIE:

És important destacar que VALID és un servei ofert per la AOC (Consorci Administració Oberta de Catalunya) i nosaltres des de PDS només consumim la seva passarel·la d'autenticació delegada, però no som responsables del *Identity Provider* destí.

Per a l'autenticació delegada a VALID, des de la PDS només podem recuperar els següents atributs del login que ha fet el ciutadà:

- Codi d'usuari: també és el DNI
- Nom i cognoms
- Correu electrònic

Aquests paràmetres poden figurar com a *claims* en el token d'accés resultant d'una autenticació a VALID. Però en el cas de VALID, no tenim un sistema de Control de Recursos per assignar atributs addicionals als ciutadas.

I a més, com el mateix cas que amb GICAR: NO es poden incorporar paràmetres addicionals com a *claims* en el token, o fer invocacions al motor de regles.

De nou, en aquest cas, la solució consisteix en disposar d'un *Client* en un *Realm* a banda, i establir un intercanvi del token d'accés obtingut amb el login a VALID, en un flux ACF, per un nou token al Realm de la iniciativa, amb un flux CCF.

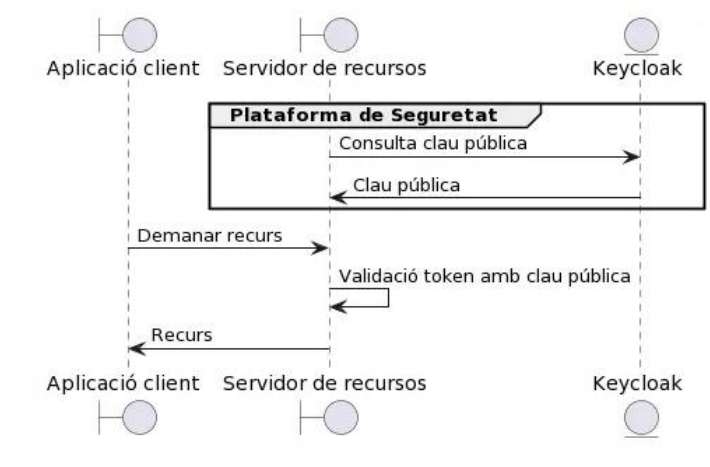
2.5 Validació del token d'accés i autorització en els backends destí

Per cada petició que ha d'atendre, la iniciativa no necessita invocar explícitament a la PDS per verificar si el token adjuntat és correcte (no està caducat, ni manipulat).

Seguint l'estàndard OIDC, per aquesta validació del token en els sistemes adherits, la PDS exposa un endpoint amb les claus públiques dels certificats amb els que signa els tokens JWT.

D'aquesta manera, l'aplicació pot verificar de forma desacoblada la validesa del token.

Existeixen implementacions de llibreries com per exemple **Spring Security** que ja realitzen aquesta verificació del token i extracció dels *claims* de forma automatitzada, configurant simplement els endpoints OIDC del Realm de la PDS.



3. Accions requerides a PDS per admetre l'adhesió (proveïdor)

Per tal de que l'adhesió sigui possible, són necessàries les següents accions tècniques, depenent dels requeriments del sistema a integrar la seva autorització amb PDS, per tal de configurar dins el Keycloak l'àmbit del que faran ús els sistemes i serveis de l'adhesió:

- Connectivitat per xarxa.
- Creació del *Realm*.
- Creació de *Client* ACF o CCF, segons el cas
- Configuració de l'endpoint on recollir el certificat.
- Configuració del plugin del Protocol Mapper, per si calen paràmetres o validacions addicionals en el token d'accés.
- Configuració del plugin de l'Event Listener, per enregistrar auditoria d'activitat de la PDS a la Plataforma d'Auditoria HES.
- Configuració del plugin de Keycloak Metrics, per enregistrar mètriques d'activitat per la monitorització interna del sistema.

Segons la ubicació del sistema d'informació que consumeix els serveis de la PDS, es pot arribar a aquests per adreçament d'Anella de Salut o adreçament públic d'Internet:

Entorn	URL Keycloak	IP Anella Salut	IP Pública
INT	https://integracio.pds.hes.catsalut.intranet.gencat.cat	146.219.255.134	-
PRE	https://preproduccio.pds.hes.catsalut.gencat.cat	146.219.253.140	83.247.168.89
PRO	https://pds.hes.catsalut.gencat.cat	146.219.252.134	83.247.135.246

4. Detall de peticions de token d'accés segons el tipus de flux OIDC

4.1 Login amb client ACF

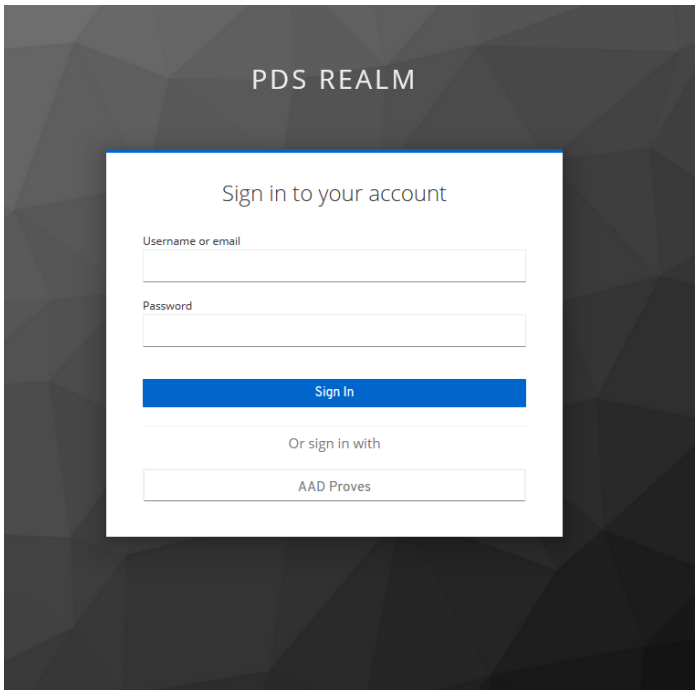
Per fer login a un *Client* ACF prèviament creat al *Realm* del Keycloak amb un usuari donat d'alta, l'aplicació ha de fer dos passos:

Petició GET a l'adreça següent:

```
{{URL_PDS}}/auth/realms/{{Realm}}/protocol/openid-
connect/auth?client_id={{ClientACF}}&response_type=code&redirect_uri={{URLRedireccio}}
```

- **{{Realm}}** és el nom del Realm al que pertany el client ACF sobre el que es vol fer login.
- **{{ClientACF}}** és el nom del client ACF al qual es vol fer login.
- **{{URLRedireccio}}** és la URL de l'aplicació on Keycloak ha de fer redirecció una vegada l'usuari hagi introduït les seves dades d'autenticació.

Aquesta petició retornarà una pàgina de login pròpia de la PDS on l'usuari haurà d'introduir les seves credencials (usuari i contrasenya). Aquest és l'estil per defecte de la pantalla que proporciona Keycloak:



4.1.1 Personalització dels estils de la pantalla de login

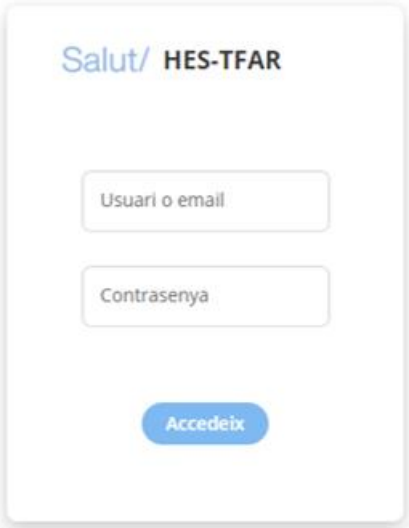
És possible personalitzar els estils i literals d'aquesta pantalla anterior, afegint nous elements, icones o literals, per tal d'adaptar-la a les necessitats de l'eina que s'adhereix. Consisteix en definir un Tema de Keycloak i facilitant-lo a l'equip de PDS per a que ho incorpori en el producte, i posteriorment fent la configuració a la PDS, aconseguim aquesta pantalla personalitzada:

https://www.keycloak.org/docs/latest/server_development/#creating-a-theme

Es pot veure un exemple de codi d'estils de Keycloak en el següent enllaç de Github:

https://github.com/keycloak/keycloak/blob/main/themes

Un cop activat el tema al Realm de la PDS, al visualització de les pantalles de login o logout seran les aplicades per la configuració:



Si les dades introduïdes són correctes, Keycloak farà una redirecció a la URL indicada a l'atribut ***redirect_uri***, passant per paràmetre un codi d'estat (**state**) i el **code** de la sessió, que s'ha de fer servir al segon pas.

Petició POST a l'adreça següent:

```
{{URL_PDS}}/auth/realms/{{Realm}}/protocol/openid-connect/token
```

- **{{Realm}}** és el nom del Realm al que pertany el client ACF sobre el que es vol fer login.

A la capçalera cal afegir Content-Type amb valor application/x-www-form-urlencoded.

I al contingut Form de la petició cal informar:

- **client_id**: Nom del client ACF creat al Realm del Keycloak.
- **client_secret**: Secret del client ACF, en cas de configurar-se com a tipus *Confidential*
- **grant_type**: Literal "authorization_code"
- **code**: Code rebut com a paràmetre en la redirecció del pas anterior.
- **redirect_uri**: URL on Keycloak ha fet la redirecció

Aquesta petició retornarà l'access_token que l'aplicació farà servir en posteriors peticions a recursos protegits i un refresh_token per poder renovar la sessió quan caduqui (punt 4.3).

La PDS també permet fer el flux ACF utilitzant PKCE (Proof Key for Code Exchange) configurat al client en el Realm de Keycloak. D'aquesta manera, s'acompanya la petició amb un code_challenge en format S256.

https://www.authlete.com/developers/pkce/

4.2 Generar token d'accés amb client CCF

Per fer login a un client CCF, cal fer una petició POST a l' adreça següent:

```
{{URL_PDS}}/auth/realms/{{Realm}}/protocol/openid-connect/token
```

- **{{Realm}}** és el nom del Realm al que pertany el client CCF sobre el que es vol fer login.

A la capçalera cal afegir ***application/x-www-form-urlencoded*** al valor del *Content-Type*.

I al contingut *Form* de la petició cal informar:

- **client_id** : Nom del client CCF creat al Realm del Keycloak.
- **client_secret**: Credencial configurada al client CCF.
- **grant_type**: Literal "**client_credentials**".
- Resta de camps que enviem com a part dels claims que s'utilitzaran per generar el token. Més detall a la secció 4.4.

Aquesta petició retornarà l'**access_token** que l'aplicació farà servir en posteriors peticions a recursos protegits. I un **refresh_token** per renovar la sessió quan caduqui (punt 4.3).

4.3 Petició Refresh per obtenir un nou token

Per obtenir un nou token a partir d'una sessió ja autenticada, cal fer ús del **refresh_token** que es rep a la resposta del login, i fer una petició POST a l'adreça següent:

```
{{URL_PDS}}/auth/realms/{{Realm}}/protocol/openid-connect/token
```

- **{{Realm}}** és el nom del Realm al que pertany el client sobre el que es va fer login.

A la capçalera cal afegir **application/x-www-form-urlencoded** al valor del Content-Type.

I al contingut Form de la petició cal informar:

- **client_id** : Nom del client de la qual la sessió es vol renovar.
- **client_secret**: La clau secret de l'autenticació. Només per a clients CCF, no cal pels ACF.

- **grant_type**: Literal “refresh_token”.
- **refresh_token:** El valor del refresh_token obtingut quan es va fer l’autenticació.

Aquesta petició retornarà un nou **access_token** per accedir a recursos, així com un nou **refresh_token** per a futures extensions de la sessió.

4.4 Petició User Info per obtenir els atributs de la sessió

Per obtenir la informació guardada en sessió de l'User Info, cal fer una petició GET a l'adreça següent:

```
{{URL_PDS}}/auth/realms/{{Realm}}/protocol/openid-connect/userinfo
```

- **{{Realm}}** és el nom del Realm al que pertany el client CCF sobre el que es vol fer login.

A la capçalera cal afegir:

- **Authorization:** Literal “Bearer” seguit del valor del access_token obtingut quan es va fer l’autenticació.

Aquesta petició retornarà el **client_id** que s'utilitzi per obtenir el access_token i també retorna un **userinfo** amb tota la informació sol·licitada per recollir mitjançant aquesta petició

4.5 Exemple de petició de logout

Per finalitzar la sessió que la PDS ha obert davant una petició ACF o CCF, és necessari fer una request de logout al Realm, indicant en quin client s’ha generat aquesta sessió associada als tokens d’accés:

```
GET {{URL_KEYCLOAK}}/auth/realms/{{Realm}}/protocol/openid-connect/logout
```

Per aquesta request de logout, la PDS necessita tenir aquest identificador de la sessió, i té dues maneres d’obtenir aquesta dada:

- Si el flux OIDC a la PDS s’ha generat enviant el scope=openid, la resposta haurà estat un token d’accés (access_token) un token de refresc (refresh_token) i un tercer token identificador de sessió (id_token) amb un payload més reduït, que simplement serveix com ID de la sessió. En aquest cas, en la petició de logout s’envia aquest token, i la PDS ja podrà determinar de quina sessió dins el client s’està demanant fer el logout. Per tant, aquest logout es fa directament.

  id_token_hint: token identificador sessió obtingut en el flux OIDC amb scope=openid

- Si no tenim aquest id_token identificador de la sessió, llavors la PDS presentarà davant la request de logout un frontend demanant la confirmació d’aquest logout, i aprofitarà per extreure aquest ID de sessió de la cookie. En cas de confirmar l’operació, llavors ja es procedeix a fer el logout.

Per altre banda, també cal enviar en la request de logoutels paràmetres:

- **post_logout_redirect_uri**: URL on cal redirigir un cop alliberada la sessió a la PDS

- **client_id**: identificador del client en la petició original de token

En el cas de GICAR o VALID, la post_logout_redirect_uri ha de ser la del propi Identity Provider, ja que cal fer una neteja de cookies en el navegador de l’usuari. Per tant:

GICAR:

- https://preproduccio.autenticaciogicar4.extranet.gencat.cat/idp/logoutgicar.jsp
- https://autenticaciogicar4.extranet.gencat.cat/idp/logoutgicar.jsp

VALID:

- https://preproduccio.passarelagicarvalid.idp1-gicar.gencat.cat/corpvalididp/Custom/Logout
- https://passarelagicarvalid.idp1-gicar.gencat.cat/corpvalididp/Custom/Logout

Per últim, és important recordar que els diferents backends que autoritzen els seus serveis amb tokens de la PDS, habitualment ho fan de forma autònoma. És a dir: no estan contínuament en cada petició anant a confirmar amb la PDS que el token segueix vigent. Per aquest motiu, es demana que la vigència del access_token no sigui molt gran en temps (màxim 15 minuts), doncs es podria donar el cas d'un token associat a una sessió de la que ja s'ha fet logout, i que els backends on s'utiltiza aquest token no siguin conscients d'aquesta operació de logout.

Al cap i a la fi, cada backend només verifica que el token és vàlid (no s'ha manipulat ni generat a mà) i que no està caducat.

La única manera de controlar que un token a més està associat a una sessió ja feta logout, és fent un introspect del token a la pròpia PDS, o fent una crida al endpoint del UserInfo de la PDS, o demanant un nou access_token en base a un refresh_token existent.

4.6 Paràmetres addicionals a la petició per generar el token.

A l'hora d'autenticar-se, segons les necessitats de l'entitat que s'adhereix, es pot afegir a la configuració les validacions i comprovacions de paràmetres obligatoris necessaris.

Malgrat des de PDS no ens ocupem de la autorització que fa cada iniciativa dels serveis publicats en els seus backends, sempre intentarem mantenir un criteri unificat durant el moment de definir aquests paràmetres. I en cas que algun paràmetre sol·licitat ja existeixi en algun *Realm* anterior, aprofitar el mateix nom, per no generar massa diferència en els diferents *Clients* i *Realms* que mantenim dins la PDS, i també reaprofitar validacions o configuracions ja implementades.

Adjuntem, a mode d'exemple, diferents paràmetres que solen ser habituals en les peticions de configuració de Clients en la PDS.

4.6.1 Exemple paràmetres sense validació

Aquesta és una llista a mode d'exemple. Cada client pot tenir els paràmetres que necessiti.

Nom del paràmetre	Descripció	Comentaris
trace_user_id	Identificardor de l'usuari	
trace_user_given_name	Nom de l'usuari	
trace_user_family_name	Cognoms de l'usuari	
module_code	Codi del mòdul	

identifier_value	Identificador/s	Aquest paràmetre es de tipus llista i permet múltiples valors separats per comes. Si s'informa, cal informar també el paràmetre *identifier_type* amb el mateix nombre d'elements.
alt_identifier_value	Identificador/s alternatius	Aquest paràmetre es de tipus llista i permet múltiples valors separats per comes. Si s'informa, cal informar també el paràmetre *alt_identifier_type* amb el mateix nombre d'elements.

4.6.2 Exemple paràmetres validats contra llistes tancades locals

Aquesta és una llista a mode d'exemple. Cada client pot tenir els paràmetres que necessiti.

Nom del paràmetre	Descripció	Exemple de valors vàlids	Comentaris
user_type	Tipus d'usuari	**PRO**, **PAC** o **ADM**	-
identifier_type	Tipus d'identificador	**DNI**, **NUMCOL**, **PASS**, **NIE**, **ECAP** o **MPI**	Aquest paràmetre es de tipus llista i permet múltiples valors separats per comes. Si s'informa, cal informar també el paràmetre *identifier_value* amb el mateix nombre d'elements.
alt_identifier_type	Tipus d'identificador alternatiu	**DNI**, **NUMCOL**, **PASS**, **NIE**, **ECAP** o **MPI**	Aquest paràmetre es de tipus llista i permet múltiples valors separats per comes. Si s'informa, cal informar també el paràmetre *alt_identifier_value* amb el mateix nombre d'elements.
role_type	Tipus de rol	**NORM**, **URG**, **TEC**	-

4.6.3 Exemple paràmetres validats contra catàlegs del Servidor Terminològic

Aquesta és una llista a mode d'exemple. Cada client pot tenir els paràmetres que necessiti.

Nom del paràmetre	Descripció	OID del catàleg	Comentaris
ep_code	Codi de l'EP	2.16.840.1.113883.4.292.10.12.5	-
up_code	Codi de l'UP	2.16.840.1.113883.4.292.10.12.6	-
center_code	Codi del centre	2.16.840.1.113883.4.292.10.12	-
center_type	Tipus de centre	2.16.840.1.113883.4.292.10.12	Es valida que el tipus indicat sigui el que correspon al centre del paràmetre *center_code*.
service_code	Codi del servei	2.16.724.4.9.40.4	-

4.7 Validació del token mitjançant la clau pública.

Cada *Realm* té associat un certificat de tipus RSA. El genèric que s'ha creat per tots els recursos associats a la plataforma utilitza l'algoritme de xifrat RS256, però Keycloak ens permet donar servei pels següents:

- RS256

- RS384
- RS512
- PS256
- PS384
- PS512

Les iniciatives adherides, com a part del seu procés d'autorització, tenen l'obligació de validar si els token que reben són vàlids. A tal efecte, cal obtenir la clau pública que Keycloak proporciona en el següent endpoint per cada *Realm*:

```
{{URL_PDS}}/auth/realms/{{Realm}}/protocol/openid-connect/certs
```

- **{{Realm}}** és el nom del *Realm* que s'ha configurat per l'adhesió.

Depenent de la tecnologia de l'aplicació, és possible que existeixen ferramentes que ajuden a fer la validació. A l'enllaç següent es poden consultar diferents llibreries amb la implementació del validador segons la tecnologia: https://openid.net/developers/jwt/

Per un detall més ample del funcionament de les signatures de seguretat web basades en JSON es pot consultar l'enllaç següent: https://www.rfc-editor.org/rfc/rfc7515

5. Apèndix

5.1 Exemple de peticions

5.1.1 Exemple petició CCF

Exemple de generació de **token** al client *ccf-client* del Realm *PdS-DUMMY*:

```
curl --request POST \
  --url                 'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY/protocol/openid-
connect/token?=' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data client_id=ccf-client \
  --data client_secret=826b3788-c902-46ba-a72d-3206afc231f4 \
  --data grant_type=client_credentials \
  --data trace_user_id=User_ID \
  --data 'trace_user_given_name=Given Name' \
  --data 'trace_user_family_name=Family Name' \
  --data user_type=ADM \
  --data 'identifier_type=DNI,NUMCOL' \
  --data 'identifier_value=73288219A,2' \
  --data alt_identifier_type=MPI \
  --data alt_identifier_value=0621cf7d-7d63-49ec-8006-8c05692ed377 \
  --data ep_code=0208 \
  --data up_code=07733 \
  --data center_type=E \
  --data center_code=E08586963 \
  --data service_code=5S089 \
  --data role_type=NORM \
  --data module_code=A001
```

Aquesta petició ens retorna l'objecte següent:

```
{
            "access_token": "...",
            "expires_in": 1800,
            "refresh_expires_in": 1800,
            "refresh_token": "...",
            "token_type": "Bearer",
            "not-before-policy": 0,
            "session_state": "ad9d8b5e-a3cd-41ac-9ba0-e4d35d5779e6",
            "scope": ""
}
```

access_token. Conté les dades de la sessió. Descodificat seria:

```
{
  "exp": 1676894537,
  "iat": 1676892737,
  "jti": "693bf427-d15d-4c16-8a9e-a7d8d9f55722",
  "iss": "https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY",
  "sub": "e0d222cb-32ef-40ac-a233-69cddb0b6338",
  "typ": "Bearer",
  "azp": "ccf-client",
  "session_state": "ad9d8b5e-a3cd-41ac-9ba0-e4d35d5779e6",
  "acr": "1",
```

```
 "realm_access": {
  "roles": [
   "myrole",
   "default-roles-pds-dummy"
  ]
 },
 "scope": "",
 "sid": "ad9d8b5e-a3cd-41ac-9ba0-e4d35d5779e6",
 "clientId": "ccf-client",
 "clientHost": "10.53.254.150",
 "access_rules": {
  "auth_type": "TBD",
  "auth_level": "TBD"
 },
 "access_info": {
  "client_id": "ccf-client",
  "client_secret": "826b3788-c902-46ba-a72d-3206afc231f4",
  "grant_type": "client_credentials",
  "trace_user_id": "User_ID",
  "trace_user_given_name": "Given+Name",
  "trace_user_family_name": "Family+Name",
  "user_type": "ADM",
  "ep_code": "0208",
  "up_code": "07733",
  "center_type": "E",
  "center_code": "E08586963",
  "service_code": "5S089",
  "role_type": "NORM",
  "identifier": [
   {
    "type": "DNI",
    "value": "73288219A"
   },
   {
    "type": "NUMCOL",
    "value": "2"
   }
  ],
  "alt_identifier": [
   {
    "type": "MPI",
    "value": "0621cf7d-7d63-49ec-8006-8c05692ed377"
   }
  ]
 },
 "clientAddress": "10.53.254.150"
}
```


refresh_token: Conté les dades per estendre la sessió. Descodificat seria:


```
{
 "exp": 1676894537,
 "iat": 1676892737,
 "jti": "c009621a-5070-4aaf-b875-51b821aef9ae",
 "iss": "https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY",
 "aud": "https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY",
 "sub": "e0d222cb-32ef-40ac-a233-69cddb0b6338",
 "typ": "Refresh",
 "azp": "ccf-client",
 "session_state": "ad9d8b5e-a3cd-41ac-9ba0-e4d35d5779e6",
```

```
  "scope": "",
  "sid": "ad9d8b5e-a3cd-41ac-9ba0-e4d35d5779e6"
}
```

expires_in i **refresh_expires_in**: Segons de validesa del token d'accés i de refresc, respectivament.

Exemple de la petició de **Refresh**:

```
curl --request POST \
  --url 'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY/protocol/openid-connect/token?=' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data client_id=ccf-client \
  --data client_secret=826b3788-c902-46ba-a72d-3206afc231f4 \
  --data grant_type=refresh_token \
  --data refresh_token=...
```

Que retorna un objecte amb nous *access_token* i *refresh_token*.

5.1.2 Exemple petició User Info

Exemple de generació de token al client ccf-userinfo del Realm PdS-DUMMY.

Petició del pas 1:

```
curl --request POST \
  --url 'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY/protocol/openid-connect/token?=' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data client_id=ccf-client \
  --data client_secret=826b3788-c902-46ba-a72d-3206afc231f4 \
  --data grant_type=client_credentials \
  --data scope=openid \
  --data ep_code=0208 \
  --data up_code=07733 \
  --data center_type=E \
  --data center_code=E08586963 \
  --data service_code=5S089 \
  --data role_type=NORM \
  --data module_code=A001
```

Petició del pas 2:

```
curl --request GET \
  --url https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY/protocol/openid-connect/userinfo\
  --header 'Authorization: Bearer access_token-generat-Pas1' \
```

Aquesta petició ens retorna l'objecte següent:

```
{
   "sub": "f53d4f4c-39ca-4882-bcee-2e305740d76e",
   "clientHost": "10.53.254.150",
   "clientAddress": "10.53.254.150",
   "client_id": "ccf-client",
   "userinfo": {
      "service-code": "5S089 ",
      "module-code": "A001 "
   }
}
```

Els paràmetres que apareixen en l' objecte de retorn, cal indicar-los per sortir en la resposta d' aquesta petició. Estan prèviament configurats.

5.1.3 Exemple petició ACF

En el moment de fer aquesta documentació, no podem oferir cap exemple de petició ACF que inclogui la code_challenge amb PKCE.

Exemple de generació de **token** al client *acf-client* del Realm *PdS-DUMMY*.

Petició del pas 1:

```
curl --request GET \
  --url                'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY/protocol/openid-connect/auth?client_id=acf-client&response_type=code&redirect_uri=http%3A%2F%2F2700-is3.conso.com%2Fpds-acf-frontend%2F' \
  --header 'Content-Type: text/html;charset=utf-8'
```

A la finestra que surt, s'introdueixen les dades d'usuari i en fer la redirecció s'obté el **code**:

```
2457cc0a-0976-4f1e-a05a-c73819be8c5e.e996fd8a-2208-40b0-8c8c-36d99d0c858b.aba970ff-e1b5-41b2-b3ae-92a8d4a2ca91
```

Petició del pas 2:

```
curl --request POST \
  --url                'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY/protocol/openid-connect/token?=' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data        code=2457cc0a-0976-4f1e-a05a-c73819be8c5e.e996fd8a-2208-40b0-8c8c-36d99d0c858b.aba970ff-e1b5-41b2-b3ae-92a8d4a2ca91 \
  --data grant_type=authorization_code \
  --data trace_user_id=0000001 \
  --data trace_user_given_name=usuari \
  --data user_type=PRO \
  --data 'identifier_type=D,C' \
  --data 'identifier_value=1234,5678' \
  --data alt_identifier_type=E \
  --data alt_identifier_value=9 \
  --data ep_code=0208 \
  --data up_code=07733 \
```

```
  --data center_type=H \
  --data center_code=E08586963 \
  --data service_code=SC \
  --data role_type=RT \
  --data sector_code=SC \
  --data 'trace_user_family_name=Family Name' \
  --data client_id=acf-client \
  --data redirect_uri=http://2700-is3.conso.com/pds-acf-frontend/
```

Aquesta petició ens retorna l'objecte següent:

```
{
            "access_token": "...",
            "expires_in": 1800,
            "refresh_expires_in": 1800,
            "refresh_token": "...",
            "token_type": "Bearer",
            "not-before-policy": 0,
            "session_state": "e996fd8a-2208-40b0-8c8c-36d99d0c858b",
            "scope": ""
}
```

access_token. Conté les dades de la sessió. Descodificat seria:

```
{
  "exp": 1676897258,
  "iat": 1676895458,
  "auth_time": 1676895343,
  "jti": "1f4e3f6a-5316-47ea-bbb4-a28aabcdb8ac",
  "iss": "https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY",
  "aud": "account",
  "sub": "f6daabc4-a1db-4e10-bce3-bd9fb6d9d6ad",
  "typ": "Bearer",
  "azp": "acf-client",
  "session_state": "e996fd8a-2208-40b0-8c8c-36d99d0c858b",
  "acr": "0",
  "allowed-origins": [
    "*"
  ],
  "realm_access": {
    "roles": [
      "myrole"
    ]
  },
  "resource_access": {
    "account": {
      "roles": [
        "manage-account",
        "manage-account-links",
        "view-profile"
      ]
    }
  },
  "scope": "",
  "sid": "e996fd8a-2208-40b0-8c8c-36d99d0c858b",
  "email_verified": false,
  "access_rules": {
    "auth_type": "TBD",
```

```
    "auth_level": "TBD"
  },
  "name": "Usuari Prova",
  "access_info": {
    "client_id": "acf-client",
    "grant_type": "authorization_code",
    "trace_user_id": "0000001",
    "trace_user_given_name": "usuari",
    "trace_user_family_name": "Family+Name",
    "user_type": "PRO",
    "ep_code": "0208",
    "up_code": "07733",
    "center_type": "H",
    "center_code": "E08586963",
    "service_code": "SC",
    "role_type": "RT",
    "sector_code": "SC",
    "identifier": [
      {
        "type": "D",
        "value": "1234"
      },
      {
        "type": "C",
        "value": "5678"
      }
    ],
    "alt_identifier": [
      {
        "type": "E",
        "value": "9"
      }
    ]
  },
  "preferred_username": "usuari",
  "given_name": "Usuari",
  "family_name": "Prova",
  "email": "usuari@viewnext.com"
}
```


refresh_token: Conté les dades per estendre la sessió. Descodificat seria:


```
{
  "exp": 1676897258,
  "iat": 1676895458,
  "jti": "aa888c91-7ca4-4235-a484-91c6913a028d",
  "iss": "https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY",
  "aud": "https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY",
  "sub": "f6daabc4-a1db-4e10-bce3-bd9fb6d9d6ad",
  "typ": "Refresh",
  "azp": "acf-client",
  "session_state": "e996fd8a-2208-40b0-8c8c-36d99d0c858b",
  "scope": "",
  "sid": "e996fd8a-2208-40b0-8c8c-36d99d0c858b"
}
```


expires_in i **refresh_expires_in**: Segons de validesa del token d'accés i de refresc, respectivament.

5.1.4 Exemple petició Refresh Token

Per qualsevol flux OIDC obert amb la PDS en que tinguem un refresh-token vigent:

```
curl --request POST \
  --url                 'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY/protocol/openid-
connect/token?=' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data client_id=acf-client \
  --data grant_type=refresh_token \
  --data refresh_token=...
```

Es retorna un objecte amb nous *access_token* i *refresh_token*.

5.1.5 Exemple petició Logout

En el moment de fer aquesta documentació, no disposem encara de cap exemple de petició de logout.

5.1.6 Alta d'usuaris

Exemple de generació de token al client **admin-cli** del Realm PdS-DUMMY.

Petició del pas 1:

```
curl --request POST \
  --url                 'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY/protocol/openid-
connect/token?=' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data client_id=admin-cli \
  --data client_secret=<client_secret> \
  --data grant_type=client_credentials\
```

Petició del pas 2:

```
curl --request POST \
--url 'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/admin/realms/PdS-DUMMY/users' \
--Authorization 'Bearer Token: Token obtingut en pas 1' \
--Body '
{
   "firstName": "Nom Usuari",
   "lastName": "Cognom Usuari",
   "username": "NickUsuari",
   "enabled": "true",
   "credentials": [
      {
         "type": "password",
         "value": "PassUsuari",
         "temporary": false
      }
   ]
}
'
```

Si el servei tornés un error, hi ha que validar que el client té ben configurat el Client Scope (amb el rol “manage-users”), vore el “Manual d’explotació” apartat 4.3.7.

5.1.7 Llistat d'usuaris

Exemple per llistar els usuaris d'un Realm, en aquest cas del Realm PdS-DUMMY fent servir el client admin-cli.

Petició del pas 1:

```
curl --request POST \
  --url                 'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY/protocol/openid-
connect/token?=' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data client_id=admin-cli \
  --data client_secret=<client_secret> \
  --data grant_type=client_credentials\
```

Petició del pas 2:

```
curl --request GET \
--url 'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/admin/realms/PdS-DUMMY/users' \
--header 'Authorization: Bearer <Token obtingut en pas 1>' \
```

El resultat es el llistat d'usuaris del Realm en el següent format:

```
[{
  "id": "efe6f057-35fa-4e5c-9551-1339515d2dab",
  "createdTimestamp": 1689589732550,
  "username": "nickusuari",
  "enabled": true,
  "totp": false,
  "emailVerified": false,
  "firstName": "Nom Usuari",
  "lastName": "Cognom Usuari",
  "disableableCredentialTypes": [],
  "requiredActions": [],
  "notBefore": 0,
  "access": {
            "manageGroupMembership": true,
            "view": true,
            "mapRoles": true,
            "impersonate": false,
            "manage": true
  }
}]
```

5.1.8 Esborrat d'usuaris

Exemple per esborrar els usuaris d'un Realm, en aquest cas del Realm PdS-DUMMY fent servir el client admin-cli.

Petició del pas 1:

```
curl --request POST \
  --url                'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/PdS-DUMMY/protocol/openid-connect/token?=' \
  --header 'Content-Type: application/x-www-form-urlencoded' \
  --data client_id=admin-cli \
  --data client_secret=<client_secret> \
  --data grant_type=client_credentials\
```

Petició del pas 2:

```
curl --request GET \
--url 'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/admin/realms/PdS-DUMMY/users' \
--header 'Authorization: Bearer <Token obtingut en pas 1>' \
```

Del resultat del pas 2 obtenim el id de l'usuari que volem esborrar:

```
[{
  "id": "7fdb0d00-621c-4bb6-a47d-c6a2d86db0f1",
  ...
}]
```

Petició del pas 3:

```
curl --request DELETE \
--url     'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/admin/realms/PdS-DUMMY/users/<id     usuari obtingut al pas 2>' \
--header 'Authorization: Bearer <Token obtingut en pas 1>' \
```

5.2 Petició de tiquet SAML

Per fer petició a un client SAML prèviament creat i configurat el *Realm* de PDS, sol·licitant-ho en el procés d'adhesió, amb un usuari donat d'alta, l'aplicació ha de fer el següent pas:

Petició GET a l'adreça següent:

```
{{URL_PDS}}/auth/realms/{{Realm}}/protocol/saml/clients/{{Client-url-name}}
```

- **{{Realm}}** és el nom del Realm al que pertany el client SAML sobre el que es vol fer la petició.
- **{{Client-url-name}}** es el nom del fragment d' URL al qual es farà referència al client quan es vol realitzar l' inici de sessió únic iniciat per IDP.

Existeix la possibilitat de definir els **SAML Attributes** que ha d'incloure els tiquets SAML. En aquest cas, com la crida per obtenir el token de PDS ha de ser de tipus GET, només podem ampliar aquesta informació a través de *Requests Headers*.

Actualment el balancejador que serveix el tràfic de PDS fa un tractament *case-insensitive* del nom d'aquests *headers*, per tant, no tindríem manera de fixar aquests atributs alternant majúscules i minúscules. Per aquest motiu, el format de la capçalera que cal enviar és

- Header Name: SAML_**NomAtribut**
- Header Value: SAML_**NomAtribut**####**ValorAtribut**

D'aquesta manera. la PDS genera el parell clau-valor del **SAML Attribute** en base al valor de la capçalera:

```
<saml:Attribute
       Name="NomAtribut"
       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
       <saml:AttributeValue
              xmlns:xs=http://www.w3.org/2001/XMLSchema
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:type="xs:string">ValorAtribut</saml:AttributeValue>
</saml:Attribute>
```

En un futur, si resolem aquest inconvenient de les capçaleres *case-insensitive*, potser no serà necessari fer aquest tractament amb els separadors "####". Tanmateix, ho mantindrem dins de la PDS com retrocompatible pels sistemes que ja facin les peticions d'aquesta manera.

Exemple de la petició:

```
curl --request GET \
  --url       'https://integracio.pds.hes.catsalut.intranet.gencat.cat/auth/realms/HC3-SAML/protocol/saml/clients/hc3-saml-salut' \
  --header 'Authorization: Basic < usuari:contrasenya codificat a base64>' \
  --header 'SAML_FirstFamilyName: 'SAML_FirstFamilyName####Nombre_Apellidos' \
  --header 'SAML_AccessType: 'SAML_AccessType ####MUNI' \
  ...
```

El resultat és un document **SAML Response** codificat en Base64. En cas de descodificar el contingut podem veure l'estructura XML amb alguns dels camps més rellevants:

<samlp:Response>	Root principal del document XML
<saml:Issuer>	Realm de la PDS que ha generat el tiquet SAML
<saml:Assertion>	Contingut del tiquet SAML
<dsig:Signature>	Signatura utilitzada per al tiquet, i clau pública
<saml:Subject>	Credencials facilitades per identificar al sol·licitant del tiquet SAML

<saml:Conditions>	Validesa del tiquet (configurable a PDS)
<saml:AttributeStatement>	Llista d'atributs SAML inclosos en el tiquet, en base als valors enviats per les capçaleres de petició.

5.3 Referències

- Keycloak
  https://www.keycloak.org/
- OpenId Connect
  https://openid.net/connect/
- JWT Decoder
  https://jwt.io/
- Client Credentiasl Flow OIDC
  https://www.rfc-editor.org/rfc/rfc6749#section-4.4
- Authorization Code Flow OIDC:
  https://www.rfc-editor.org/rfc/rfc6749#section-4.1
- RefreshToken OIDC:
  https://www.rfc-editor.org/rfc/rfc6749#section-6
- Spring Security
  https://spring.io/projects/spring-security
- Autenticació híbrida GICAR i VALID:
  https://canigo.ctti.gencat.cat/plataformes/gicar/integracions/funcionalitats-autenticacio/auth-4-passarelagicarvalid/?cms=true
- GICAR
  https://ctti.gencat.cat/ca/ctti/solucions-corporatives/gestio-didentitats/gicar/
- VALID
  https://www.aoc.cat/es/serveis-aoc/valid/