SPID

Interfaccia

Integrazione ENTI

Sommario

1 Introduzione

Il documento ha lo scopo di fornire le indicazioni operative per l'utilizzo, lato client, degli attributi utente disponibili a seguito dell'autorizzazione dell'utente finale.
Il sistema utilizza il protocollo standard Oauth2 introdotto nel prossimo paragrafo.

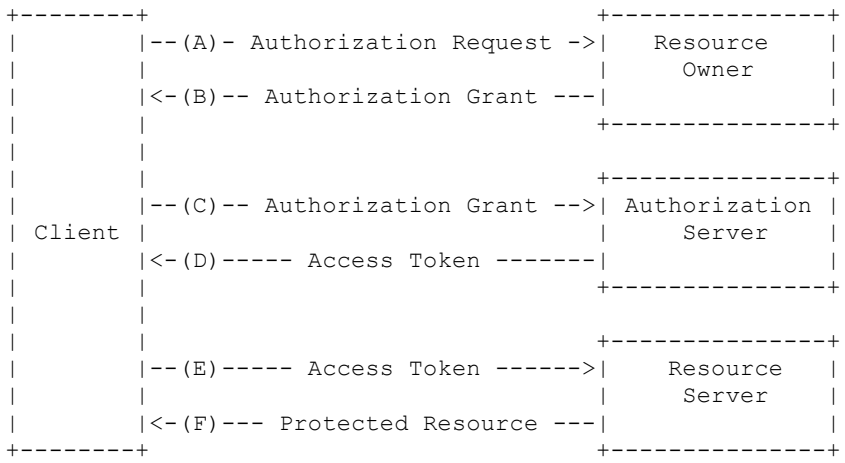
1.1 Oauth2

Il protocollo di comunicazione Open Authorization chiamato comunemente OAuth permette ad un'applicazione o ad un servizio web di gestire in modo sicuro l'accesso autorizzato ai dati sensibili. Esso è stato ideato nel 2006 da Blaine Cook e si posto come alternativa ai protocolli proprietari aperti già esistenti quali Google AuthSub, AOL OpenAuth, Yahoo BBAuth, Flickr API e tanti altri. La prima versione 1.0 rilasciata nel 2007 è stata oggetto di numerose revisioni sviluppate a seguito del recepimento delle RFC (Request For Comments) proposte dai vari esperti. La versione più recente e attualmente utilizzata del protocollo è la OAuth 2.0. Attraverso l'uso di questo protocollo viene garantito l'autorizzazione a terze parti ad accedere ad informazioni private senza condividere la password personale.
L'OAuth che garantisce un accesso delegato ad un client relativamente a delle risorse specifiche, mantenute su un server per un tempo limitato, e revocabile dall'utente che le utilizza. Tale delega deve essere autorizzata dall'utente stesso, prima di accedere alla risorsa. All'interno del protocollo possono essere individuati tre principali attori coinvolti:

- **Resource Server** ovvero il servizio che implementa la logica di business; è colui quindi che è capace di accettare dei token autorizzativi (chiamati access token) attraverso i quali è in grado di capire se chi sta effettuando la richiesta è autorizzato a farlo;

- **Client** è l'applicazione che richiede l'accesso alle risorse protette dell'utente (previa autorizzazione) per conto del proprietario della risorsa (App nativa, applicazione web, TV, Application server, ecc);
- **Resouce Owner** è l'utente che, essendo registrato ad una piattaforma del service provider vuole garantire l'accesso alle risorse personali che in esso sono presenti; è colui che concede l'accesso alla risorsa protetta e, nel caso in cui si tratti di una persona fisica, viene indicato come utente;
- **Authorization** Server che rilascia l'access token al client dopo che l'utente proprietario della risorsa ha effettuato con successo l'autenticazione e ha concesso l'autorizzazione al client;

Quando un'applicazione client tenta l'accesso ad un servizio fornito da un Resource Server, viene effettuata un handshake che consente al Resource Owner di autorizzare il client ad accedere ai propri dati o permessi (chiamati scope).



Il client è chi direttamente interagisce con il Resource Server e può essere:

- **Confidential**, client capaci di custodire in modo sicuro le loro credenziali. Ad esempio un'applicazione web eseguita su un web server è da considerarsi un client sicuro. Il questo caso il Resource Owner interagisce con il server attraverso il proprio browser. Le credenziali del client si trovano in un posto sicuro sul server e non possono essere utilizzate in dal resource owner;
- **Public**, client che non possono custodire in modo sicuro le loro credenziali (ad esempio un'applicazione nativa mobile oppure un'applicazione browser based)

1.1.1 Authorization Grant Type (grant_type)

Authorization Grant Type sono credenziali che rappresentano l'autorizzazione ottenuta da parte del proprietario di una risorsa e che vengono utilizzate dal client per ottenere un access token. La specifica del protocollo OAuth definisce 4 tipi di grant authorization dei quali verranno descritti i 2 disponibili in ambiente Linea Comune.

Code, ottenuto utilizzando un server di autorizzazione come intermediario tra il client ed il proprietario della risorsa. Infatti il client, invece effettuare una richiesta direttamente al proprietario della risorsa, si rivolge ad un server di autorizzazione che a sua volta ottiene il codice di autorizzazione dal proprietario della risorsa e lo fornisce al client. Quindi il proprietario della risorsa si autentica solo con il server di autenticazione in modo che le sue credenziali non vengano condivise con il client. Ciò garantisce un elevato livello di sicurezza in quanto l'access token è scambiato solo ed unicamente tra client ed authorization server

Implicit è una versione semplificata del caso precedente ed usato dei linguaggi di scripting oppure mobile in quanto hanno bisogno di interagire direttamente con il Resource Server. Con un flusso implicito, anziché di fornire un codice di autorizzazione, viene rilasciato al client direttamente l'access token.

1.1.2 Access Token

L'access Token è la credenziale utilizzata per accedere alle risorse protette. Si tratta di una stringa rilasciata al client con una validità limitata nel tempo (generalmente pochi minuti). Un access token fornisce un livello di astrazione elevato in quanto i diversi costrutti di autorizzazione (ad esempio username e password) vengono sostituiti con un singolo token che è interpretato e compreso dal Resource Server. Tale astrazione permette di concedere degli access token più restrittivi rispetto all'authorization grant usata per ottenerli, inoltre ciò elimina la necessità da parte del server delle risorse di comprendere una vasta gamma di metodi di autenticazione. Gli access token possono avere differenti formati, strutture e metodi di utilizzo a seconda dei requisiti di sicurezza del server delle risorse.

1.1.3 Refresh Token

Il Refresh Token viene rilasciato al client da un authorization server per ottenere un nuovo grant quando l'access token corrente è considerato invalido o è scaduto, oppure al fine di ottenere ulteriori access token con un ambito uguale o più ristretto. L'emissione di un refresh token è facoltativa ed a discrezione dell'authorization server. Qualora esso venga rilasciato dal server è incluso nell'access token relativo fornito. Il refresh token è una stringa che rappresenta l'autorizzazione concessa al client dal proprietario della risorsa. Il token denota un identificativo usato per recuperare le informazioni di autorizzazione. A differenza dell'access token, il refresh token sono destinati ad un uso esclusivo con gli authorization server e non sono mai inviati ai Resource Server.

1.2 Authorization e Resource Server Endpoint

In questa sezione sono descritti gli endpoint messi a disposizione da Linea Comune e che implementano le specifiche Oauth2.

1.2.1 Authorization Endpoint

L'authorization Endpoint risponde all'indirizzo */iam/oauth2/authorize* e la richiesta deve essere effettuata direttamente dal Resource Owner in quanto è lui che dovrà autenticarsi (CNS, SPID, username & password).
I Client possono usare i metodi HTTP GET e POST per inviare la richiesta di autorizzazione al Server di Autenticazione. Se si usa il metodo http GET, i parametri della richiesta vengono serializzati mediante URI Query String Serialization. Se si utilizza il metodo http PUT, i parametri della richiesta vengono serializzati utilizzando il Form Serialization.
I parametri in ingresso possono essere:

Nome Parametro	Obbligatorio	Descrizione
scope	SI	Lista separata da spazi con la lista degli scope desiderati. Per la lista completa vedi paragrafo 2.1.
client_id	SI	Un identificatore OAuth 2.0 Client valido per l'Authorization Server.
response_type	SI	Dipende dall'Authorization Flow utilizzato e può essere valorizzato con *code* oppure *token*.
redirect_uri	SI	L'URI di Reindirizzamento a cui verrà inviata la risposta. Questo URI deve corrispondere esattamente a uno dei valori URI di reindirizzamento per il Client preregistrato presso Linea Comune.
state	NO	Valore opaco utilizzato per mantenere lo stato tra la richiesta e il callback. Tipicamente, la mitigazione del Cross-Site Request Forgery (CSRF, XSRF) avviene vincolando crittograficamente il valore di questo parametro con un cookie del browser.

Se l'utente è autenticato viene mostrata una pagina di autenticazione con la quale autorizzare l'invio degli attributi richiesti. Dopo l'autenticazione il browser dell'utente viene inoltrato (HTTP 302) all'url presente in *redirect_uri* con valorizzati dei parametri aggiuntivi a seconda del *response_type* utilizzato e *state* valorizzato al valore precedentemente inviato.
Se l'utente risulta non autenticato viene inoltrato alla pagina di autenticazione.

Esempio di richiesta:

```
HTTP/1.1 302 Found
Location: https://identificazione.055055.it/iam/oauth2/authorize?re-
sponse_type=XXX &scope=profile%20email &client_id=clientProva
&state=af0ifjsldkj &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

Se response_type=token la risposta sarà una redirect di questo tipo;

```
https://client.example.org/?gws_rd=ssl#access_token=46cb5bdd-67bc-4c91-ba02-
f6fa04a8ae1a&expires_in=599&token_type=Bearer
```

Se response_type=code la risposta sarà una redirect di questo tipo;

```
https://client.example.org/?code=12345
```

1.2.2 Token Endpoint

Un client effettua una richiesta di Token all'indirizzo */iam/oauth2/access_token* presentando la sua Authorization Grant (sotto forma di un Authorization Code) al Token Endpoint utilizzando il valore grant_type=authorization_code, come descritto nella Sezione 4.1.3 di OAuth 2.0 [RFC6749].
Il Client invia i parametri al Token Endpoint utilizzando il metodo HTTP POST e il modulo di serializzazione come descritto nella Sezione 4.1.3 di OAuth 2.0 [RFC6749].

Nome Parametro	Obbligatorio	Descrizione
grant_type	SI	Può si desidera ottenere access e refresh token dato: *authorization_code,* *refresh_token*
code	SI se grant_type = *authorization_code*	Il codice ricevuto dall'Authorization Server.
refresh_token	SI se grant_type = *refresh_token*	Il Refresh Token.
scope	NO	Se omessi vengono ereditati quelli della richiesta originale.
redirect_uri	SI	L'URI di Reindirizzamento indicato nella richiesta all'authorization endpoint

```
POST /iam/oauth2/access_token HTTP/1.1
Host: https://identificazione.055055.it
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW grant_type=authoriza-
tion_code&code=SplxlOBeZQQYbYS6WxS bIA&redirect_uri=https%3A%2F%2Fclient.exam-
ple.org%2Fcb
```

Dopo aver ricevuto e validato una Richiesta di Token dal Client, valida ed autorizzata, l'Authorization Server restituisce una risposta di successo che include un Access Token. I parametri nella risposta di successo sono definite nella sezione 4.1.4 di OAuth 2.0 [RFC6749]. La risposta utilizza il tipo di supporto application / json.

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
Cache-Control: no-store Pragma: no-cache
{ "access_token": "SlAV32hkKG", "token_type": "Bearer", "refresh_token":
"8xLOxBtZp8", "expires_in": 3600}
```

1.2.3 TokenInfo Endpoint

Il servizio TokenInfo restituisce gli attributi utenti in base agli scope indicati nelle precedenti richieste.

Nome Parametro	Obbligatorio	Descrizione
access_token	SI	Un Access Token Valido

```
https://identificazione.055055.it/iam/oauth2/tokeninfo? access_token=46cb5bdd-
67bc-4c91-ba02-f6fa04a8ae1a
```

Di seguito un esempio di risposta:

```
{"mail":"email@example.com","scope":["mail","address","cn"],"address":"via
Roma 12 12345 X323 RM","cn":"RSSMNC80S30X323K","realm":"/","to-
ken_type":"Bearer","expires_in":453,"access_token":"46cb5bdd-67bc-4c91-ba02-
f6fa04a8ae1a"}
```

2 Utilizzo Oauth2 Linea Comune

In questo capitolo viene descritto come una applicazione, federata con Linea Comune, può recuperare gli attributi per i quali l'utente ha autorizzato la visualizzazione.

Innanzitutto è necessario richiedere a Linea Comune la generazione delle credenziali che il client dovrà utilizzare per accedere all'Authorization Server. Nella richiesta è necessario specificare la tipologia di client che può essere:

1. **Confidential**, in caso di applicazioni backend come applicazioni web seguire la documentazione al paragrafo 2.3. In questo caso il browser dell'utente non invoca mai direttamente il Resource Server ma la chiamata è mediata dall'application server.
2. **Public**, in caso di applicazioni (es native, client javascript) che interrogano direttamente un Resource Server

In entrambe i casi è necessario specificare:

- **scope** supportati (vedi paragrafo 2.1)
- **redirect_uri**, vedi 1.2.2.
- **durata dell'access token**, default 5 minuti

2.1 Scope

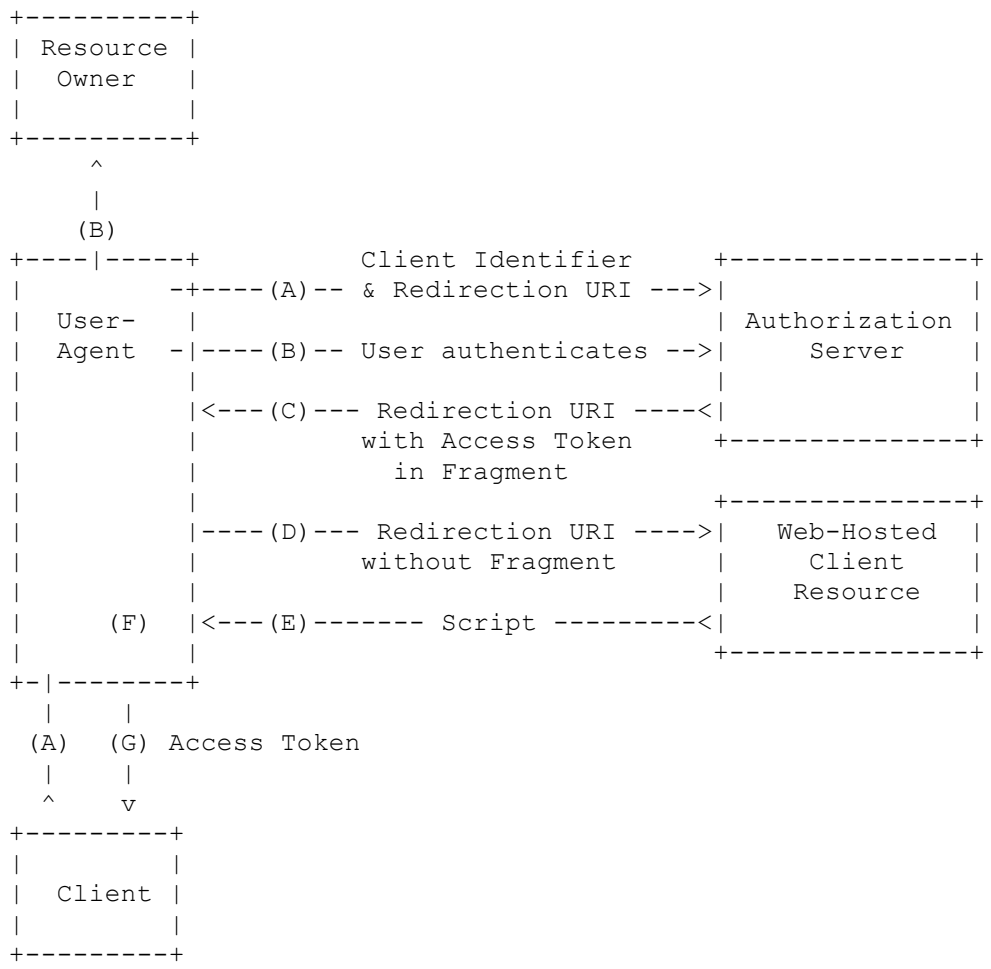
Gli Scope disponibili sono dipendenti dal tipo di autenticazione utilizzata dall'utente. Per tipologia di autenticazione SPID gli attributi sono di responsabilità diretta dall'Identity Provider SPID utilizzato.

Nome	Tipo Autenticazione	Esempio	Descrizione
cn	*	RSSMNC80S30X323K	Il codice fiscale dell'utente
spidCode	SPID		Il codice identificativo relativo all'identity provider utilizzato
name	SPID	Mario	Il nome del soggetto
familyName	SPID	Rossi	Il cognome del soggetto
dateOfBirth	SPID	1980-12-11	Data di nascita nella forma AAA-MM-GG
placeOfBirth	SPID	K321	Codice catastale comune di nascita
fiscalNumber	SPID	TINIT-RSSMNC80S30X323K	Codice Fiscale con prefisso TINIT-
gender	SPID	M	Sesso M, F
email	*	pinco@pallino.com	Email
address	SPID	via Roma 12 12345 K321 RM	Il dato restituito da SPID. Indirizzo, di norma nella forma via numero, codice postale, codice catastale comune,

			codice provincia
idCard	SPID	cartaIdentita AK1234545 comuneRoma 2010-08-19	Il dato restituito da SPID. Documento identificativo, di norma nella forma tipo carta, numero, ente rilasciato, data rilascio
mobilePhone	SPID	+393201234567	Numero di cellulare comprensivo di prefisso internazionale
peopleLuogoNascita	*	K321	Codice catastale comune di nascita
peopleNome	*	Mario	Il nome del soggetto
peopleSesso	*	M	Sesso M, F
peopleTelefono	*	+393201234567	Numero di cellulare comprensivo di prefisso internazionale
authType	*	Federation	Il tipo di autenticazione utilizzata (Federation, Cns)
authLevel	*	2	Il livello di autenticazione utilizzato (per SPID valori ammessi 1, 2, 3)
IssueInstant	SPID	2017-09-18 15:30:25.000	Data ed ora generazione asserzione SPID
AuthnContextClassRef	SPID	https://www.spid.gov.it/SpidL2	Il livello SPID di autenticazione restituito dall'asserzione di autenticazione

2.2 Autenticazione tramite Implicit Grant Flow

Quando si utilizza l'Implicit Flow, tutti i token vengono restituiti dall'Authorization Endpoint; il Token endpoint non viene utilizzato se non per ottenere un nuovo access token dato un refresh token.

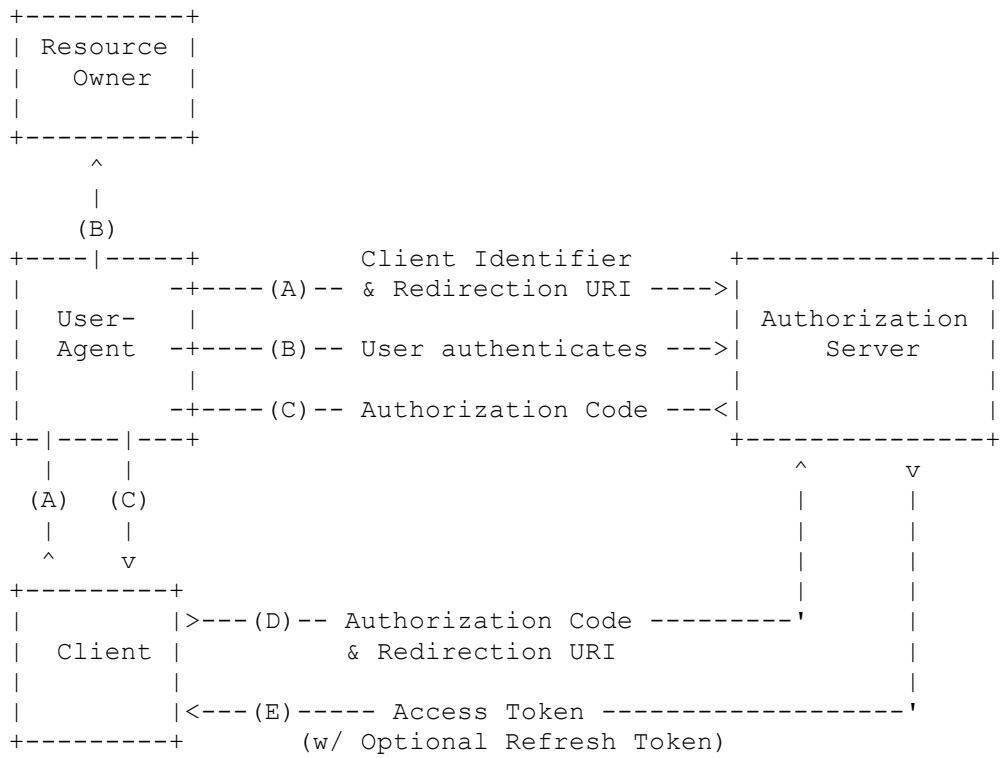


L'Implicit Flow segue le seguenti fasi:

a. Il Client prepara una richiesta di autenticazione contenente i parametri di richiesta desiderati o User-Agent invia la richiesta all'Authorization Server (vedi 1.2.1 con response_type= token);
b. L'Authorization Server richiede all'Authentication Server di autenticare l'utente ed ottiene il consenso e l'autorizzazione dell'utente finale
c. L'Authorization Server effettua una redirect dello User-Agent all'indirizzo indicato (può essere anche fittizio)
d. Lo User-Agent segue la redirect
e. Uno script oppure codice custom intercetta la redirect ed estrae l'Access Token
f. Lo User-Agent riceve l'Access Token
g. Lo User-Agent passa l'Access Token al Client

2.3 Autenticazione tramite Authorization Code Flow

In questa sezione viene descritto come eseguire l'autenticazione utilizzando l'Authorization Code Flow. Quando si utilizza questo flusso, tutti i token vengono restituiti dal Token Endpoint. Il tipico utilizzo di questo tipo di flusso è l'utilizzo in applicazioni web che interagiscono con Linea Comune lato Server.
L'Authorization Code Flow restituisce un codice di autorizzazione (Authorization Code) al Client, che può successivamente scambiarlo direttamente per un ID Token e un Access Token. Questo offre il vantaggio di non esporre alcun token all'User Agent. L'Authorization Server può anche autenticare il Client prima di scambiare l'Authorization Code per un Access Token. L'Authorization Code flow è adatto per i Client in grado di mantenere in modo sicuro un Client Secret tra loro e l'Authorization Server.



L' Authorization Code Flow Flow segue le seguenti fasi:

a. Il Client prepara una richiesta di autenticazione contenente i parametri di richiesta desiderati o User-Agent invia la richiesta all'Authorization Server (vedi 1.2.1 con response_type= code);
b. L'Authorization Server richiede all'Authentication Server di autenticare l'utente ed ottiene il consenso e l'autorizzazione dell'utente finale
c. L'Authorization Server effettua una redirect dello User-Agent all'indirizzo indicato
d. Il Client recupera il code e lo utilizza per recuperare il token dall'Authorization Server(vedi 1.2.2)
e. L'Authorization Server restituisce access e refresh token

2.3.1 Frammento di codice

Di seguito è mostrato un frammento di codice che fa uso delle librerie spring e scribejava per implementare un controller web dove l'endpoint */login* che semplicemente inoltra il browser dell'utente all'endpoint authorize di Linea Comune, e riceve il parametro code ottenuto all'endpoint */oauth_callback* il quale viene utilizzato per ricavare l'access token che a sua volta viene utilizzato per recuperare le informazioni utente.

```
@Controller
public class Oauth2Controller {
       private OAuth20Service service = new ServiceBuilder("client-id")
             .apiSecret("client-secret")
             .scope("cn mail address")
             .callback("http://localhost:8080/oauth_callback/")
             .build(LineaComuneApi.instance());

       @RequestMapping("/login")
       public void login(HttpServletResponse response) throws IOException {
         response.sendRedirect(service.getAuthorizationUrl());
    }

       @RequestMapping("/oauth_callback")
       @ResponseBody
   public String redirect(@RequestParam(value="code") String code) throws Exception{
       //Recupero Access Token dato il code
        final OAuth2AccessToken accessToken = service.getAccessToken(code);
       //Invoco il servizio tokeinfo per il recupero degli attrivuti utente
        final OAuthRequest request = new OAuthRequest(Verb.GET, LineaComuneApi.in-
stance().getUserInfoEndpoint());
         service.signRequest(accessToken, request);
         return service.execute(request).getBody();
    }
       private static class LineaComuneApi extends DefaultApi20 {
           private static class InstanceHolder {
               private static final LineaComuneApi INSTANCE = new LineaComuneApi();
           }
           public static LineaComuneApi instance() {
               return InstanceHolder.INSTANCE;
           }
           public String getUserInfoEndpoint() {
               return "https://identificazione.055055.it/iam/oauth2/tokeninfo";
           }
           @Override
           public String getAccessTokenEndpoint() {
               return "https://identificazione.055055.it/iam/oauth2/access_token";
           }
           @Override
           protected String getAuthorizationBaseUrl() {
               return "https://identificazione.055055.it/iam/oauth2/authorize";
           }
           @Override
           public OAuth2SignatureType getSignatureType() {
               return OAuth2SignatureType.BEARER_URI_QUERY_PARAMETER;
           }
       }
}
```