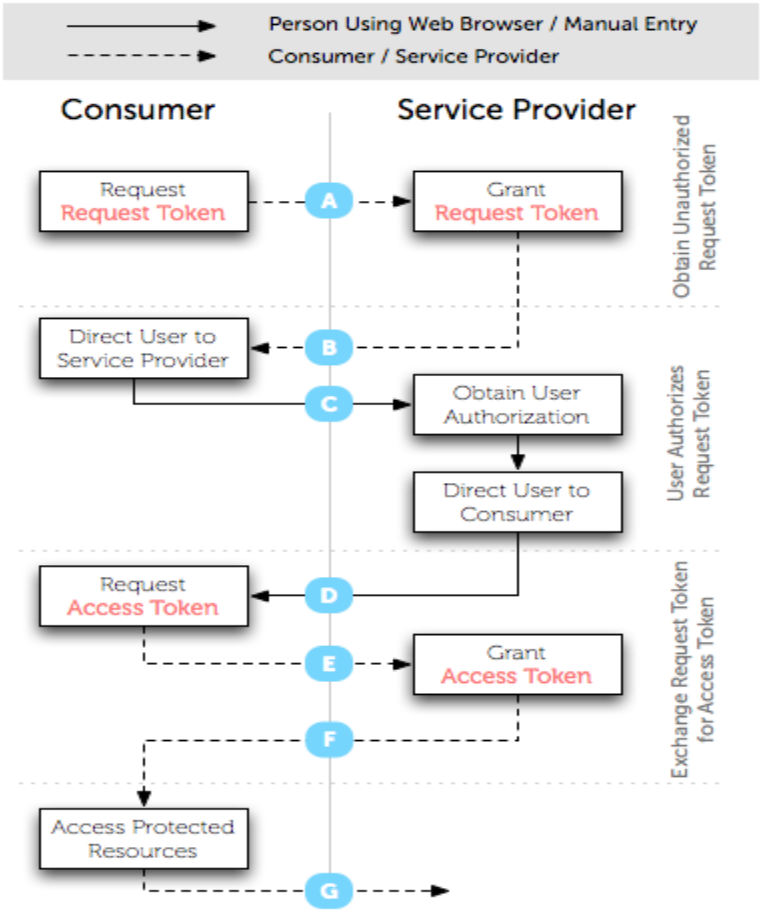
OpenAM OAuth 2.0

Il seguente documento riguarda il supporto del protocollo OAuth 2.0 da parte del sistema di identificazione di Linea Comune.
Le specifiche che riguardano il protocollo OAuth 2.0 possono essere trovate all'indirizzo http://tools.ietf.org/html/rfc6749. Il sito di riferimento è http://oauth.net/2/.
Il sistema di identificazione di Linea Comune rispetta le specifiche definite nell' **RFC 6749** con alcune eccezioni che verranno descritte più avanti ma che non vanno in alcun modo a modificare il flusso del protocollo.

Descrizione

Il protocollo OAuth 2.0 consente ad un applicativo terzo di avere un accesso limitato ad un servizio HTTP, sia per conto di un proprietario di una risorsa orchestrando un'interazione di approvazione tra il proprietario delle risorse e il servizio HTTP, sia consentendo ad un'applicazione di terze parti di ottenere l'accesso per conto proprio.
Il seguente diagramma riassume il flusso del protocollo OAuth 2.0.



Registrazione del servizio

Come previsto dall' RFC 6749 il servizio deve registrarsi all'Authorization server di Linea Comune. (vedi http://tools.ietf.org/html/rfc6749#section-2)

Ottenere l'autorizzazione

Il sistema di identificazione di Linea Comune limita il tipo di autorizzazione a quello definito "Authorization Code". (vedi http://tools.ietf.org/html/rfc6749#section-4.1)

Authorization Request
(vedi http://tools.ietf.org/html/rfc6749#section-4.1.1)

https://identificazione.055055.it/iam/oauth2/authorize

L'Authorization request è usata dal client (applicativo) per ottenere l'autorizzazione dal proprietario della risorsa mediante il re-direzionamento dello user agent sul server di autenticazione.

Il paragrafo 4.1.1 del RFC 6749 definisce i parametri obbligatori ed opzionali per la costruzione dell'Authorization URL. Rispetto a quanto definito dall' RFC il parametro **redirect_uri** diventa obbligatorio. E' possibile comunque definire in fase di registrazione del servizio una o più redirect uri.
Oltre ai parametri definiti nel paragrafo 4.1.1 il sistema di identificazione di Linea Comune prevede un ulteriore parametro, **realm**. Il parametro realm è associato al client id e viene rilasciato in fase di registrazione.

Access Token Request
(vedi http://tools.ietf.org/html/rfc6749#section-4.1.3)

https://identificazione.055055.it/iam/oauth2/access_token

L'Access Token Request è usato dal client (applicativo) per ottenere un access token. Il **redirect_uri** diventa un parametro obbligatorio perché definito nell'Authorization Request.

Accessing Protected Resources
(vedi http://tools.ietf.org/html/rfc6749#section-7)

https://identificazione.055055.it/iam/oauth2/tokeninfo

Il client (applicativo) usa l'access token per accedere alle risorse definite nell'Authorization Request mediante il parametro **scope** (vedi http://tools.ietf.org/html/rfc6749#section-3.3). Il parametro scope viene concordato e definito in fase di registrazione del servizio.

Access Token
(vedi http://tools.ietf.org/html/rfc6749#section-1.4)

L'Access Token viene rilasciato solo una volta è ha una validita per un periodo limitato di tempo.

Refresh Token

Il sistema di identificazione di Linea Comune non prevede il Refresh Token per evitare che un client (applicativo) possa chiedere un nuovo Access Token. Il client (applicativo) accede alla risorsa per un periodo di tempo limitato e deve gestire autonomamente I dati, attraverso la sessione utente per esempio.

Esempi applicativi

Di seguito sono riportati due esempi applicativi, uno in JAVA e uno in PHP, che spiegano un esempio di interazione di con il sistema di identificazione di Linea Comune. Per altri linguaggi di programmazione fare riferimento al sito http://oauth.net/2/.

JAVA

L'esempio di basa sulla libreria (https://github.com/fernandezpablo85/scribe-java). La libreria implementa un'API per tutti i maggiori OAuth 2.0 providers.
In questo esempio troviamo due classi, OpenamApi che definisce tutte le URL di interazione con il sistema di indentificazione di Linea Comune e LineaComuneExample che dimostra l'utilizzo dell'api.

OpenamApi.java

```
public class OpenamApi extends DefaultApi20 {

    private final static String DEFAULT_REALM = "/forall";

    private final static String ACCESS_TOKEN_ENDPOINT =
"https://identificazione.055055.it/iam/oauth2/access_token?
grant_type=authorization_code&realm=%s";

    private static final String AUTHORIZE_URL =
"https://identificazione.055055.it/iam/oauth2/authorize?realm=%s&client_id=
%s&redirect_uri=%s&response_type=code";

    private static final String SCOPED_AUTHORIZE_URL = AUTHORIZE_URL
            + "&scope=%s";

    private String realm;

    public OpenamApi() {
        this(DEFAULT_REALM);
    }

    public OpenamApi(String realm) {
        this.realm = realm;
    }

    @Override
    public String getAccessTokenEndpoint() {
        return String.format(ACCESS_TOKEN_ENDPOINT, this.realm);
    }

    @Override
    public String getAuthorizationUrl(OAuthConfig config) {
        // Append scope if present
        if (config.hasScope()) {
            return String.format(SCOPED_AUTHORIZE_URL, this.realm,
                    config.getApiKey(),
                    OAuthEncoder.encode(config.getCallback()),
                    OAuthEncoder.encode(config.getScope()));
        } else {
            return String.format(AUTHORIZE_URL, this.realm,
config.getApiKey(),
                    OAuthEncoder.encode(config.getCallback()));
        }
    }
```

```
    @Override
    public Verb getAccessTokenVerb() {
        return Verb.POST;
    }

    @Override
    public AccessTokenExtractor getAccessTokenExtractor() {
        return new JsonTokenExtractor();
    }

}
```

LineaComuneExample.java

```
public class LineaComuneExample {

    private static final String PROTECTED_RESOURCE_URL =
"https://identificazione.055055.it/iam/oauth2/tokeninfo";
    private static final Token EMPTY_TOKEN = null;

    public static void main(String[] args) {

        // Replace these with your own api key and secret
        String apiKey = "apiKey";
        String apiSecret = "apiSecret";
        OAuthService service = new ServiceBuilder()
                                    .provider(new OpenamApi("/forall"))
                                    .apiKey(apiKey)
                                    .apiSecret(apiSecret)

.callback("http://www.iltuoservizio.it/")
                                    .scope("cn mail sn")
                                    .build();

        Scanner in = new Scanner(System.in);

        // Obtain the Authorization URL
        System.out.println("Fetching the Authorization URL...");
        String authorizationUrl = service.getAuthorizationUrl(EMPTY_TOKEN);
        System.out.println("Got the Authorization URL!");
        System.out.println("Now go and authorize Scribe here:");
        System.out.println(authorizationUrl);
        System.out.println("And paste the authorization code here");
        System.out.print(">>");
        Verifier verifier = new Verifier(in.nextLine());
        System.out.println();

        // Trade the Request Token and Verfier for the Access Token
        System.out.println("Trading the Request Token for an Access
Token...");
        Token accessToken = service.getAccessToken(EMPTY_TOKEN, verifier);
        System.out.println("Got the Access Token!");
        System.out.println("(if your curious it looks like this: " +
accessToken.getRawResponse() + " )");
        System.out.println();

        // Now let's go and ask for a protected resource!
        System.out.println("Now we're going to access a protected
resource...");
```

```
        OAuthRequest request = new OAuthRequest(Verb.GET,
PROTECTED_RESOURCE_URL);
        service.signRequest(accessToken, request);
        Response response = request.send();
        System.out.println("Got it! Lets see what we found...");
        System.out.println();
        System.out.println(response.getBody());

    }

}
```

PHP

L'esempio si basa sulla libreria (https://github.com/fkooman/php-oauth-client). Per installare la libreria fare riferimento alla documentazione.

L'esempio deve essere pubblicato su un server web e si basa su due file php, index.php (il punto di ingresso) e callback.php (che fa parte della callback uri o redirect uri).

Index.php

```
<?php

require_once 'autoload.php';

$apiUri = "https://identificazione.055055.it/iam/oauth2/tokeninfo?
access_token=";

$clientConfig = new \fkooman\OAuth\Client\ClientConfig(
    array(
        "authorize_endpoint" =>
"https://identificazione.055055.it/iam/oauth2/authorize?realm=/forall",
        "client_id" => "client_id",
        "client_secret" => "client_secret",
        "token_endpoint" =>
"https://identificazione.055055.it/iam/oauth2/access_token?
grant_type=authorization_code&realm=/forall",
    "credentials_in_request_body" => true,
        "redirect_uri" => "http://www.iltuservizio.it/callback.php"
    )
);

$tokenStorage = new \fkooman\OAuth\Client\SessionStorage();
$httpClient = new \Guzzle\Http\Client();
$api = new fkooman\OAuth\Client\Api("foo", $clientConfig, $tokenStorage,
$httpClient);

$context = new \fkooman\OAuth\Client\Context("servizio1", array("cn", "sn",
"mail"));

$accessToken = $api->getAccessToken($context);
error_log(print_r($accessToken, true));
if (false === $accessToken) {
    /* no valid access token available, go to authorization server */
    header("HTTP/1.1 302 Found");
    header("Location: " . $api->getAuthorizeUri($context));
    exit;
}

try {
    $client = new \Guzzle\Http\Client();
    $bearerAuth = new
\fkooman\Guzzle\Plugin\BearerAuth\BearerAuth($accessToken->getAccessToken());
    $client->addSubscriber($bearerAuth);
    $response = $client->get($apiUri .
$accessToken->getAccessToken())->send();
    header("Content-Type: application/json");
```

```
    echo $response->getBody();
} catch
(\fkooman\Guzzle\Plugin\BearerAuth\Exception\BearerErrorResponseException $e)
{
    if ("invalid_token" === $e->getBearerReason()) {
        // the token we used was invalid, possibly revoked, we throw it away
        $api->deleteAccessToken($context);
        $api->deleteRefreshToken($context);
        /* no valid access token available, go to authorization server */
        header("HTTP/1.1 302 Found");
        header("Location: " . $api->getAuthorizeUri($context));
        exit;
    }
    throw $e;
} catch (\Exception $e) {
    die(sprintf('ERROR: %s', $e->getMessage()));
}
```

callback.php

```
<?php

require_once 'autoload.php';

$clientConfig = new \fkooman\OAuth\Client\ClientConfig(
    array(
        "authorize_endpoint" =>
"https://identificazione.055055.it/iam/oauth2/authorize?
realm=/forall&redirect_uri=http://www.google.it/",
        "client_id" => "client_id",
        "client_secret" => "client_id",
        "token_endpoint" =>
"https://identificazione.055055.it/iam/oauth2/access_token?
grant_type=authorization_code&realm=/forall",
    "credentials_in_request_body" => true,
    "redirect_uri" => "http://www.iltuservizio.it/callback.php"
    )
);

try {
    $tokenStorage = new \fkooman\OAuth\Client\SessionStorage();
    $httpClient = new \Guzzle\Http\Client();
    $cb = new \fkooman\OAuth\Client\Callback("foo", $clientConfig,
$tokenStorage, $httpClient);
    $cb->handleCallback($_GET);

    header("HTTP/1.1 302 Found");
    header("Location: http://www.iltuservizio.it/index.php");
    exit;
} catch (\fkooman\OAuth\Client\AuthorizeException $e) {
    // this exception is thrown by Callback when the OAuth server returns a
    // specific error message for the client, e.g.: the user did not
authorize
    // the request
    die(sprintf("ERROR: %s, DESCRIPTION: %s", $e->getMessage(),
$e->getDescription()));
} catch (\Exception $e) {
    // other error, these should never occur in the normal flow
    die(sprintf("ERROR: %s", $e->getMessage()));
}
```