Aloha Age Verification Technical Documentation

Glossary

- **Aloha Profile.** A product providing an account that unlocks additional features such as synchronization, age verification, and other functionalities. Implemented as a separate service.
- **Age-Verification.** The process of verifying a user's age.
- **Publisher.** An adult website or any other 18+ website that requires age-verification.
- **Authorization.** The process of logging into Aloha Profile.

Introduction

Age-Verification in Aloha Profile is a service designed to check the user's age.
Technically, it is implemented as an OAuth service provider.

Workflow

The publisher places a button on their website that redirects the user to Aloha Profile.
The following flow occurs:

1. **Authorization and Age-Verification**
   The user must log in to Aloha Profile and complete the verification process.
   Re-verification is not required if the user has already successfully completed it.
2. **Return to the Publisher's Website**
   After successful age-verification, Aloha Profile automatically redirects the user to the publisher's `callback_url` with a temporary-authorization token. Aloha Profile also provides the publisher with information on whether the user is of legal age. Caching this information is recommended to prevent redundant requests to Aloha Profile.
3. **Using Aloha Profile for Authorization**
   Publishers can not only receive user age information via Aloha Profile but also use it to implement an authorization process similar to Google OAuth on other websites.

Aloha Profile supports the OpenID Connect (OIDC) protocol and implements the Authorization Code flow. Detailed article.

Integration

Operational Aspects. The first step in integration is creating an OAuth client.
The publisher needs to provide the following information:

- `return_url` : used to redirect the user in case of errors
- `redirect_url` : an array of allowed URLs for redirection after verification.
  HTTPS scheme is mandatory.

Then, on the Aloha Profile side, an OAuth client is created with the following values:

- `client_id` : unique client identifier
- `client_secret` : client secret key

Scopes

Supported OAuth scopes:

- `openid` – required scope ensuring the generation of an id_token
- `age` – scope ensuring the return of the user's age-verification result

Authorization Code Flow

Step 1: Redirect the user to Aloha Profile: https://alohaprofile.com/auth/oauth2/

Query parameters:

- `client_id` : the OAuth client value obtained earlier from Aloha Pro
- `redirect_uri` : the redirect URL to which Aloha Profile will return the result. Must be one of the allowed redirect URLs for the OAuth client
- `response_type` : set to `code`
- `scope` : `openid age`

Example Request:

```
curl --location 'https://alohaprofile.com/auth/oauth2?
client_id=publisher_client_id&redirect_uri=https%3A%2F%2Fpublisher.com%2Foauth2%2Fcallback&response_type=code&scope=openi
d+age
```

Example code for a URL generation function (TypeScript):

```
const baseUrl = 'https://alohaprofile.com';
const clientId = '...';
const redirectUrl = '...';

function generateAlohaIdUrl() {
   const url = new URL(`${baseUrl}/auth/oauth2`);

   const params = new URLSearchParams({
      client_id: clientId,
      redirect_uri: redirectUrl,
      response_type: 'code',
      scope: 'openid age',
   });

   url.search = params.toString();

   return url.toString();
}
```

Detailed Documentation: https://openid.net/specs/openid-connect-core-1_0.html#AuthRequest

Step 2: Obtaining a Temporary-Authorization Token

The verification results will be returned to the `redirect_uri` with the following query parameters:

- `code` : temporary-authorization code used to obtain a token
- `iss` : Aloha Profile URL

Example of a request received from AlohaID:

```
{redirect_uri}?code=NaqUyXIqzPQzWmq93_5rjHyf1ZqfNXngTbBjpfuVXtC&iss=https%3A%2F%2Faloha-id.com%2Fauth%2Foauth2
```

Detailed Documentation: https://openid.net/specs/openid-connect-core-1_0.html#AuthResponse

Step 3: Obtaining a JWT Token with User Data

After receiving the temporary-authorization token, the publisher must send a request to the Aloha Profile backend to obtain a JWT token.

The request `Content-Type` must be `application/x-www-form-urlencoded`

Form submission URL: `/auth/oauth2/token`

Fields:

- `code` : temporary-authorization code obtained in the previous step
- `client_id` : value received from Aloha Profile
- `client_secret` : value received from Aloha Profile
- `grant_type` : set to `authorization_code`
- `redirect_uri` : the same `redirect_uri` value that was provided in Step 1

Example of a request received from AlohaID:

```
curl --location 'https://alohaprofile.com/auth/oauth2/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'code=eVNGdjmK3G_cxrlll9Pp5qw2vSZzjMvozMahP-J7qil' \
--data-urlencode 'client_id=00000000-0000-0000-0000-000000000000' \
--data-urlencode 'client_secret=1111111111111111' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode 'redirect_uri={redirect_uri}
```

Specification Description: https://openid.net/specs/openid-connect-core-1_0.html#TokenRequest

Example Response:

```
{
  "sub": "3653162",
  "is_adult": true,
  "at_hash": "YP-qICVWC68YZwhN2583TA",
  "aud": "00000000-0000-0000-0000-000000000000",
  "exp": 1697812887,
  "iat": 1697809287,
  "iss": "https://aloha-id.com/auth/oauth2"
}
```

Specification Description: https://openid.net/specs/openid-connect-core-1_0.html#CodeIDToken

The `is_adult` field will contain information about whether the user is of legal age.
The other fields follow the standard.

Example code for handling a request from Aloha Profile and sending a request to obtain the `id_token`

(TypeScript/Express):

```
const express = require('express')
const jwt = require('jsonwebtoken');

const baseUrl = 'https://alohaprofile.com';
const clientId = '...';
const clientSecret = '...';
const redirectUrl = '...';

const app = express();

app.get('/callback', async (req, res) => {
  const code = req.query.code;

  // receive the token
  const response = await fetch(`${baseUrl}/auth/oauth2/token`,
   {
     method: "POST",
     headers: {
      "Content-Type": "application/x-www-form-urlencoded",
     },
     body: new URLSearchParams({
      code
      client_id: clientId,
      client_secret: clientSecret,
      grant_type: "authorization_code",
      redirect_uri: redirectUrl,
     }),
   }
  );

  // get id_token
  const { id_token } = await res.json();
  // parse id_token
  const { is_adult } = jwt.decode(id_token);

  // ...
});

app.listen(80, () => {
    console.log(`Example app listening on port 80`);
});
```

Errors

In case of an error, the user will be redirected to the `return_url`.