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.
- **Authorization code.** A temporary code returned to the publisher after successful authorization or age verification. The publisher exchanges this code for tokens.

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. **Redirect to Aloha Profile**
   The publisher sends the user to Aloha Profile using an OpenID Connect authorization request.
2. **Age verification**
   Aloha Profile checks whether the user already has a saved age-verification result. If not, the user starts Aloha camera-based age verification.
3. **Fallback ID verification**
   If Aloha camera verification cannot complete, the user can retry or continue with fallback ID verification through Yoti, our partner.
4. **Optional profile saving**
   If verification succeeds, the user may save the result to an Aloha Profile and reuse it across publisher sites. The user may also skip saving and return to the publisher with a one-time verification result.
5. **Return to the publisher's website:**
   After successful age verification, Aloha Profile redirects the user to the publisher's `redirect_uri` with an authorization code. The publisher exchanges the code for tokens and receives information on whether the user is of legal age.
6. **Using Aloha Profile for authorization**
   Publishers can use Aloha Profile both to receive age-verification results and 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:

- `redirect_urls` : an array of allowed URLs for redirection after authorization or verification.
  The `redirect_url` sent in the OAuth request must exactly match one of these URLs. HTTPS is required for production redirect URLs.

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 Profile
- `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`
- `state` recommended opaque value generated by the publisher and returned unchanged in the callback. Use it to protect against CSRF and match the callback to the original request

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&state=publisher_state'
```

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',
      state: crypto.randomUUID(),
   });

   url.search = params.toString();

   return url.toString();
}
```

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

Step 2: Obtain the authorization code

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

- `code` : temporary authorization code used to obtain tokens
- `iss` : Aloha Profile issuer URL
- `state` : the same value sent by the publisher in Step 1, if provided

Example of a request received from Aloha Profile:

```
{redirect_uri}?
code=NaqUyXIqzPQzWmq93_5rjHyf1ZqfNXngTbBjpfuVXtC&iss=https%3A%2F%2Falohaprofile.com%2Fauth%2Foauth2&state=publisher_state
```

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

Step 3: Obtain tokens with user data

After receiving the authorization code, the publisher must send a request to the Aloha Profile backend to obtain tokens.

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

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

Fields:

- `code` : 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=111111111111111111' \
--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:

```
{
  "access_token": "ylzl6IMldWgG5uA1nSXSoRD5B_loCtdKYkE1Ca39hww",
  "expires_in": 3600,
  "id_token": "eyJ...Z4Q",
  "scope": "openid age",
  "token_type": "Bearer"
}
```

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

`id_token` is an OAuth session token containing information about the user's age-verification status.

Example contents of `id_token` :

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

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

The `is_adult` field contains information about whether the user is of legal age. The other fields follow the OpenID Connect standard.

The `sub` value may be either an Aloha Profile ID when the result is saved to a profile, or a guest verification subject for one-time verification. Publishers should use `is_adult` as the age-verification result and must validate the `id_token` signature, issuer, audience, and expiration before trusting it.

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

const app = express();

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

  // Validate `state` against the value saved before redirecting the user.

  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,
    }),
  });
  const { id_token } = await response.json();

  // Verify the id_token signature, issuer, audience, and expiration before using it.
  const { is_adult } = await verifyIdToken(id_token);

  // ...

});

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

});
```

Errors

In normal OAuth error cases, Aloha Profile redirects the user to the provided `redirect_uri` with standard OAuth error parameters, for example `error` and optional `error_description`

Common errors include:

- `access_denied` : the user cancelled, failed verification, or declined required steps
- `invalid_request` : the OAuth interaction or submitted verification data is invalid
- `server_error` : an internal error occurred

If the OAuth interaction has expired before Aloha Profile can safely continue the redirect, Aloha Profile shows a local error page.