Purpose

Apps and digital experiences using the Wyng Profiles SDK for personalization can integrate with existing user registration and authentication mechanisms on your website or mobile app. This is required for use cases like Preference Centers, where a consumer in an authenticated session securely views and updates private profile attributes. This document summarizes our recommended approach.

For developers, helpful prior context is familiarity with the Wyng Profiles API and SDK; documentation can be found here: https://developers.wyng.com/docs/wyng-profiles/intro.

Relevant use cases

Some common use cases covered in this guide are:

- Integrating Wyng-powered Preference Centers with site or app login.
- Embedding Wyng experiences that access private profile attributes in your site or app.
- Using Wyng Profiles SDK to access private profile attributes from your site or app.

What this guide does not cover:

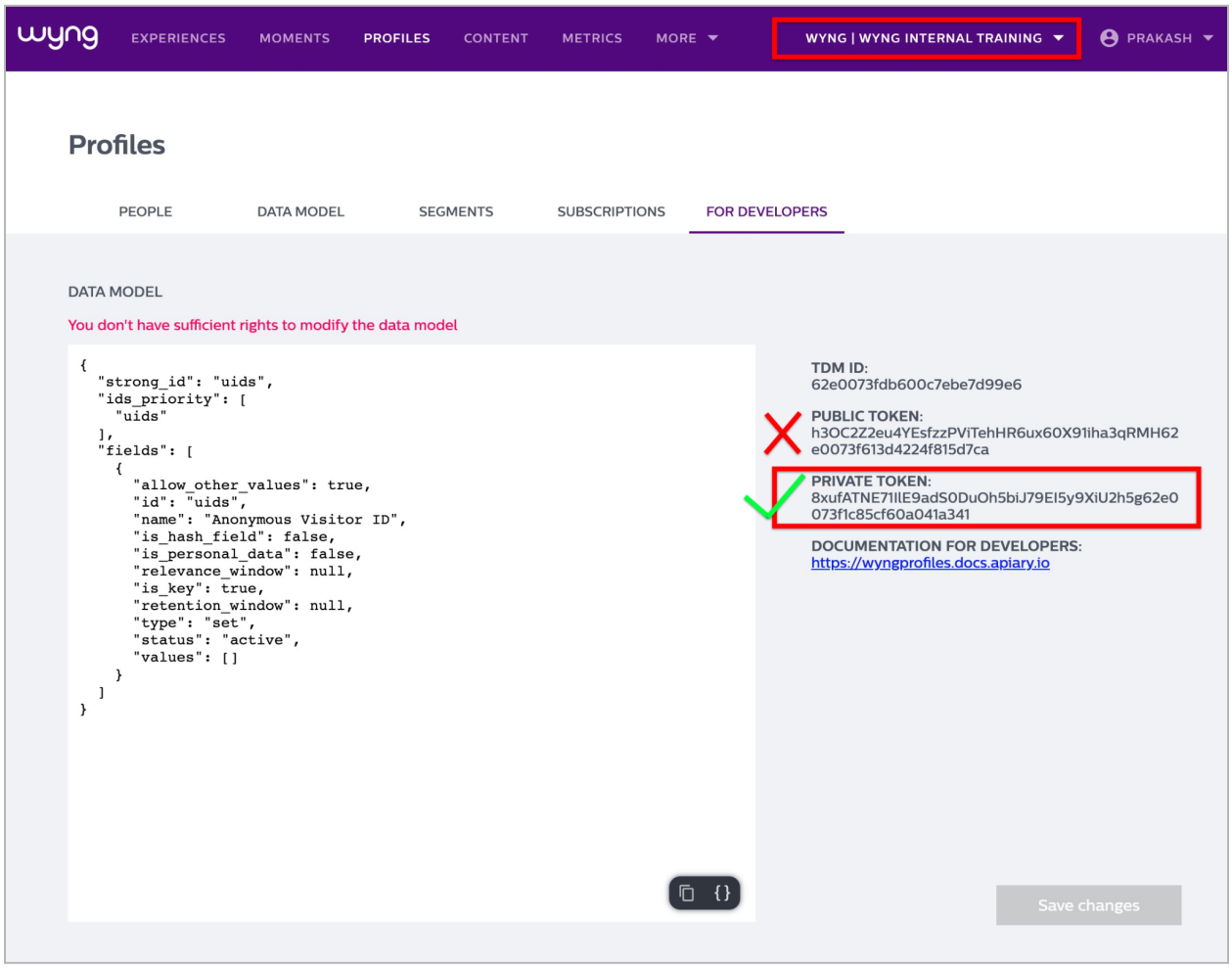
- Wyng Profiles SDK has a number of capabilities to allow personalization of unauthenticated and/or anonymous visitor sessions. This guide does not review or discuss those capabilities, and is focused on personalization in authenticated sessions.

A current, non-trial Wyng account is required for access to Wyng APIs and SDKs.

How to integrate an authenticated user session

Follow these suggested steps to integrate trusted identity between your user registration and login mechanism and Wyng:

1. Get a private token for Profiles API. This can be found by logging into Wyng, using the brand account selector in the top-right to select your production account, navigating to Profiles > For Developers, and copying the private token. ***This token has full privileges and should never be exposed publicly.***



2. Make a server-side API call to look up the Wyng profile ID for an authenticated consumer. For cases where a consumer is accessing a Preference Center or other content on a client website or mobile application that requires access to private data in the consumer's Wyng profile, the client CMS or application server should make a server-side API call to `/profiles/lookup` with a key attribute that uniquely identifies the target profile.

In most implementations, the key attribute configured as the Known User Identifier in Identity Resolution settings should be used for look ups, as this ID governs profile merging and guarantees uniqueness.

If the key attribute requires normalization to exactly match the representation in the profile, the API client is responsible for normalization.

Example:

```
GET /profiles/lookup?email=olivia.smith@example.com
```

If a profile matches the provided key, the Wyng profile ID will be returned, e.g.:

```
5f68bf46030feaa8f4b5bb7b
```

If no profile matches the key, an empty response will be returned.

Documentation:
https://developers.wyng.com/docs/wyng-profiles/profiles-api#profileslookup

3. Generate a short-lived access token for the matched profile. If a matching profile was found, configure your CMS, application server, or equivalent to make a server-side API call using the private token to `/profiles/{id}/identify` to get a JSON Web Token (JWT). The JWT is a secure, short-lived access token for that profile.

Example:

```
POST /profiles/5f68bf46030feaa8f4b5bb7b/identify
```

The access token will be generated and returned, e.g.:

```
{
   "jwt": "eyJ0...V2SA"
}
```

Documentation:
https://developers.wyng.com/docs/wyng-profiles/profiles-api#profilesprofile_ididentify

Plan for token lifespan and refresh

The generated access token has a specific expiration time, generally 30 minutes. This can be checked by inspecting the JWT. Optionally, the lifespan of the token can be customized when it is generated, to any desired value. The token can also be refreshed at any time, by regenerating a new token using the `/identify` endpoint.

To ensure that a valid token is available when consumers access their profile data, recommended practice is to re-generate a token each time a consumer visits any page on your website that requires access to their profile data, and customize the lifespan of the token to match the duration of the authenticated session on your website.

4. Surface identity and credentials to front-end code. Configure your CMS, application server, or equivalent to make both the lookup key and JWT available to front-end code on your site or mobile application in a session object. For example:

```
const userData = {
    "key_field": "email",
    "email": "olivia.smith@example.com",
    "jwt": "eyJ0...V2SA"
}

sessionStorage.setItem("userData", JSON.stringify(userData));
```

If no matching profile was found in step 2, this means that no profile yet exists for this consumer. In this case, the lookup key should still be made available to front-end code on your site in a session object, and the JWT field in the session object should be null:

```
const userData = {
    "key_field": "email",
    "email": "olivia.smith@example.com",
    "jwt": null
}

sessionStorage.setItem("userData", JSON.stringify(userData));
```

For the case that no matching profile is found, in the front-end code of the Preference center or other experience that needs access to profile data, a new profile with default settings can be created using the `identify` and `upsert` methods of the Profiles SDK.

A `userData` session object should be set for all cases where a user is authenticated. If there is no `userData` session object, front-end Preference center or other code shoulf assume the user accessing the content is not authenticated, and provide some default instruction, like a call-to-action to login.

5. Authenticate the Profiles SDK session to enable full access to private data. In the front-end code that needs access to profile data, the key field and JWT credential can be used to identify and authenticate the user session to the Profiles SDK. Once the session is authenticated, subsequent SDK methods called in that session will have full access to all the data in the profile.

Documentation:
https://developers.wyng.com/docs/wyng-profiles/profiles-sdk#identify
https://developers.wyng.com/docs/wyng-profiles/profiles-sdk#authenticate

Example:

```
window.wyng.profiles.identify( "email": "olivia.smith@example.com"
);
window.wyng.profiles.authenticate( "eyJ0...V2SA" );
window.wyng.profiles.get().then(fields => { console.log(fields);
});
```

More resources and support

- Developer docs and tutorials: https://developers.wyng.com/
- User docs: https://help.wyng.com/
- 24x7 support for users is available by email, at support@wyng.com

Technical support for developers is available on Slack; to get access, reach out to Wyng support or your account manager.

Version

This is version 1.1 updated 4-Nov-2023