Wyng Tech Guide:

Planning and Using Identifiers with Wyng Profiles API

Purpose

Proper selection and use of identifiers are essential to effective privacy-first personalization.

This guide summarizes commonly used identifiers, reviews how Wyng handles identity resolution, and suggests best practices for identifiers when building and using zero-party data profiles. When implementing Wyng, we recommend following the best practices outlined here.

In this guide:

1. Identifier Roles and Types

Identifiers play multiple roles including identifying repeat visitors to a site, associating profiles with individual people, unifying data from past anonymous sessions with known user identity when it is learned, and stitching together profiles that originate from a person using different devices.

The Wyng Profiles Data Model supports this using *Multi ID*: which provides support for multiple anonymous and known ID types, including anonymous visitor IDs, email addresses, phone numbers, and other IDs such as logged-in customer IDs and Google Analytics user IDs.

The difference between anonymous and known ID types is important, because known user IDs correspond to individuals, and may be used to reliably recognize the same individual across devices, browsers, or apps, and are the preferred way to integrate zero-party-data into your existing martech stack.

2. Anonymous Visitor Identifiers

An anonymous visitor ID represents a unique browser/device or app/device and is created and assigned to a visitor on their first visit to a site. This ID is used to identify returning visitors over multiple sessions from the same browser/device or app/device, but not across different devices, apps, or browsers. Anonymous visitor IDs are generated automatically and contain no personal information. A common example is the Google Analytics Client ID (`_ga`).

In the past, anonymous visitor IDs were generally implemented with third-party cookies, but with the growing awareness of and emphasis on personal privacy, third-party cookies have fallen out of favor, and are rapidly becoming ineffective. In privacy-centered site architectures, third-party cookies are being replaced by site-specific anonymous IDs, commonly implemented using first-party cookies or browser local storage.

The Wyng Profiles SDK generates and assigns a site-specific anonymous visitor ID on the first visit to a site, and stores it in browser local storage as `uid`, e.g., `730e2cf0cab64e3a978625cb1bb35705`. Each new `uid` creates a corresponding visitor profile in Wyng Profiles, where data shared by the visitor over multiple sessions can be collected securely. When an individual person uses multiple browsers, apps, or devices to access the site, those profiles can be merged via a process called *profile stitching* — this uses a known user identifier (like `email`) to map multiple `uids` to a single profile. For more information on profile stitching, see the Identity Resolution section below.

Other anonymous visitor IDs or cross-channel visitor IDs that are already being generated for site visitors may be used in parallel with the Wyng `uid`, such as the Google Analytics User-ID. These can be added to the Wyng Profiles Data Model using the Wyng Profiles Data Model Editor, and the relative priority of each ID type can be set.

Impact of Safari's Intelligent Tracking Prevention on identifying returning visitors

With ITP 2.3 on Safari browsers, Apple caps the lifetime of data in browser local storage to 7 days. This change was made to crack down on use of local storage to circumvent Apple's intended privacy protections limiting cross-site tracking.

Practically, this means that returning website visitors will be assigned a new anonymous visitor ID by the Wyng Profiles SDK if more than 7 days have passed since their previous visit.

To identify returning anonymous visitors using Safari over time frames longer than 7 days, we recommend using server-side first-party cookies, or HTTP cookies. Wyng Profiles SDK can be configured to use any existing HTTP cookie for anonymous visitor identification. For each website visitor, if the specified HTTP cookie exists, the SDK will sync the anonymous visitor ID with it. Additionally, for clients using Wyng-hosted landing pages on a custom domain, Wyng servers will automatically set a long-lived HTTP cookie for this purpose.

3. Known User Identifiers

A known user ID is information that uniquely identifies an individual customer or person across devices, browsers, and apps. The value of a known user ID should be unique to each individual customer or site visitor, and may contain personally identifiable information (PII), such as `email_address`, or `phone_number`. Other known user IDs do not contain PII, for example, `customer_id`, `member_id`, or a `hashed_email_address`.

In Wyng Profiles, known user IDs may be added to the Data Model and configured as keys using the Profiles Data Model Editor. Once configured, known user IDs for a current site visitor can be declared using a front-end SDK call to `wyng.profiles.identify`, or included in calls to `wyng.profiles.upsert`. Alternatively, known user IDs may be collected directly from site visitors using any form or visual questionnaire published using the Wyng Microexperience Studio.

For cases where multiple IDs are available for the current site visitor, the desired priority order in which IDs should be matched to profiles is controlled in the Profile Data Model configuration, using the setting `ids_priority`.   This is a list of all IDs in the Data Model, ordered by match priority. For example, if the Data Model is configured with the Wyng anonymous visitor ID (`uid`) plus two known user IDs, `email` and `customer_number`, and `ids_priority` setting is `["customer_number", "email", "uid"]`, then for any site visitor, the Profiles API will first try to match a profile using customer_number if possible, and if there was no match, try to match by email, and finally, by uid.

When known user IDs are declared via SDK, form, or visual questionnaire, here is how they are processed:

1. The Profiles API tries to match a profile. All available IDs are considered, along with the Wyng anonymous visitor ID (`uid`). Order of precedence for evaluating the IDs is controlled by the `ids_priority` defined in the data model.
2. If no profile matches any of the IDs, a new profile is created.
3. If any of the IDs are not already present in the profile, they are added or updated.
4. If any of the declared IDs is a known user identifier, the Profiles API checks if any other profiles exist with those IDs, and if so, unifies them by the profile stitching process.

When using Wyng to collect a known ID directly from a user (e.g. via a form), Wyng Profiles supports Flex ID, whereby SDK functions can be used to automatically pass existing IDs into any experience, so that already-identified users don't need to submit their ID since it's already known (e.g. users who clicked from an email campaign, or users who are already logged-in).

4. Pseudonymization

In some cases, there may be a need to use sensitive PII to identify known users, such as a bank account number, social security number, or driver's license number. Processing of such data is regulated internationally and in the U.S., and must be treated with care. Where sensitive PII must be used to identify known users, as a best practice, it should be pseudonymized before processing: for example, `hashed_bank_account_number`.

The Wyng Terms of Service prohibits processing of sensitive PII, including private financial and healthcare data; any such data must be pseudonymized via a secure hash before being processed or stored by Wyng.

To simplify and standardize pseudonymization, Wyng Profiles API includes native hashing of designated attributes using secure hashes. A mandatory hashing policy can be configured on any field using the Data Model Editor. When that field is processed via an SDK call or a Wyng form, the supplied value will be hashed before it is processed. To prevent issues with data integrity, mandatory hashing can only be enabled at the time a new field is added to the Profiles Data Model, and, once enabled for a field, the policy cannot be disabled.

5. Mobile Device Identifiers

Mobile apps may have access to special IDs that identify a particular mobile device, for example, IDFA on iOS, or GPS ADID on Android. Unlike anonymous visitor IDs, mobile device IDs are consistent, across all sites / browsers / apps that a person uses on their mobile device. This makes them a useful mechanism to identify returning visitors, when available.

However, while mobile device IDs share some characteristics with known user IDs (like consistency across sites / browsers / apps), they are closer in function to anonymous visitor IDs, because they do not permit identification of an individual person across devices, and they change over time, when a person switches phones, or chooses to reset their IDFA or GPS ADID. Furthermore, the availability of mobile device IDs depends on user settings and the default settings controlled by the mobile OS are becoming more restrictive over time: Apple, for example, is transitioning IDFA from opt-out, to opt-in.

6. Profile Object Identifiers

Generally speaking, customer databases maintain a unique primary key for direct access to profile records. The primary key is used by applications accessing the customer database to directly lookup or update specific profile records. Profile records in any customer database may consist of public and private data about a customer or individual person, e.g., identifiers, needs and preferences, first-party data, and transaction history.

In Wyng, profile records are managed by the Wyng Profiles API. The API generates a new profile OID when a new site visitor is identified. This may be triggered by a front-end SDK call to `wyng.profiles.identify` or `wyng.profiles.upsert`, or by a site visitor submitting a form configured using the Wyng Microexperience Studio. The profile OID is generally used behind the scenes, whereas anonymous visitor IDs or known user IDs are used in API calls, forms, or visual questionnaires, to identify site visitors.

7. Identity Resolution

Identity resolution is the process of unifying data shared by an individual person across multiple sessions, browsers, apps, and devices. Wyng Profiles API has native support for identity resolution via cross-session unification and profile stitching, to automate this process.

Cross-Session Unification

When a person visits your site for the first time, Wyng Profiles SDK generates an anonymous visitor ID (`uid`) and includes it in calls to the Profiles API. This creates a new profile, a distinct record in your Profiles Data Model instance where data shared by this site visitor over time can be stored and modeled securely.

When the person returns to your site, the Wyng Profiles SDK uses the `uid` to identify the same profile, and progressively enrich the profile with any new data shared by the user. In this way, data is unified across multiple sessions, even for an anonymous visitor.

When the person eventually shares a known user ID (e.g. `email`) or a known user ID is declared using the Profiles SDK (e.g., `customer_id` after the person logs into the site) this ID enriches the profile, upgrading it to a known user profile. All data shared by the person during previous anonymous sessions is retained, and now unified with the known user ID and any additional data shared in the current session. This is an example of *cross-session unification* in action.

Profile Stitching

One person may have multiple anonymous profiles in your Data Model instance, one for each device (or browser, or mobile app) used to access your site. Over time, as the person shares information from each device, the corresponding profiles are enriched, with each profile providing a partial view of this person.

When the person eventually shares the same known user ID, e.g. `email`, from two of those devices, we have an opportunity to unify those previously separate profiles. Profile stitching is the process of unifying two previously separate profiles, so that any more information shared by the person in the future—from either device—will enrich the unified profile.

In Wyng, known user identifiers in the data model control profile unification. If two profiles have the same value for a known user identifier, they will be unified. Commonly used known user identifiers include `email`, `phone`, `customer_id` and `member_id`.

During profile stitching, if the profiles contain overlapping attributes, the Profiles API uses attribute-level timestamps to automatically preserve the most up-to-date value for each attribute.

Profile stitching is triggered in real-time, at the moment the Profiles API determines that the same known user identifier (`email` in this example) is associated with a second profile. After unification, all `uids` and other user identifiers from either profile, are merged into the unified profile. The unified profile is immediately available for personalization across devices.

Identity Resolution Scenarios

New Anonymous Visitor: The Anonymous Visitor ID (`uid`) does not match an existing profile, so a new one is created, including any data shared by the visitor.

Returning Anonymous Visitor: The `uid` matches an existing profile, so the new data shared by the visitor enriches the existing profile. This is an example of cross-session unification.

Returning Anonymous Visitor shares an email to become a new Known User: The `uid` matches an existing profile, and an `email` is supplied, but no visitor profile has that `email`. The existing profile is enriched with the `email`. This is an example of cross-session unification.

Returning Known User: The `uid` matches an existing profile, with an `email`. Any new data shared by the visitor enriches the existing profile. This is an example of cross-session unification.

Returning Anonymous Visitor shares an email that matches a Known User on Different Device: The `uid` matches an existing profile with no `email`. However, the `email` supplied matches a different existing profile--as a result, the two profiles are stitched together.

Anonymous Visitor clicks through a coded email link to become a new Known User: A person who shared information previously in anonymous sessions receives an email with a click-through link. The link has `customer-id` encoded in a URL parameter and, on click-through, a code snippet on the website finds the `customer-id` and declares it with an SDK call to `wyng.profiles.identify`. The `uid` matches an existing profile, which is enriched with the `customer-id`. This is an example of cross-session unification.

8. Identity Verification

In some cases, having a user self-report an email (or phone number) is not enough — there is a need to verify the identity of the site visitor. For example, a brand may need to verify identity before allowing a site visitor to see and update their preferences.

Requiring the user to login to the site to access their information is one solution, but in certain scenarios that approach either will not work, or is not an acceptable user experience, for example, in an e-commerce user journey where requiring a login early hurts performance, or if login is not available on the site.

For those cases, identity verification by sending a one-time-password by email or text message, is a best practice. The one-time-password or OTP user flow is familiar to consumers, trusted, and lower friction than a login (people don't need to choose or remember a password).

Wyng has native support for identity verification using one-time-passwords in any form or visual questionnaire created using the Wyng Microexperience Studio. When ID Verification is enabled, Wyng will immediately send an OTP via email or SMS, which the site visitor can enter to confirm their identity and proceed.