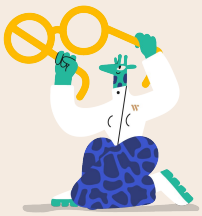
Wunderkind mailchimp

How Wunderkind Integrates With Mailchimp

The Mailchimp integration with Wunderkind will enable us to collect and send new email addresses to your Mailchimp, verify subscription status before sending an email, and update a user's subscriber status if they unsubscribe.

This guide includes an overview of everything that Wunderkind can and cannot support, considerations, requirements placed on us by Mailchimp, and more.



Wunderkind requires UI access to Mailchimp in order to build and test our integration with your Mailchimp instance.

Table of Contents:

1. Authentication

We use a Private API Key in the header of the API call.

Specifically, we'll require a **Mailchimp Data Center**, which we append to the Private API Key in the header of our API calls as well as include in the URL path for our API calls. The Data Center can be found as part of your account URL. For example, if the URL is https://us6.admin.mailchimp.com/ then the Mailchimp Data Center is us6.

2. Writing New Subscribers

Upon email submission, Wunderkind will write new subscribers to Mailchimp through the API call below.

We can write new subscribers in the following ways:

- Adding a new subscriber to a list
  - A list ID is required.
  - We cannot add subscribers “globally.”
- Optionally include categories and interests for further email segmentation.
  - These must already exist on the list for us to add contacts to them.
- Optionally include additional properties (eg “name”).
- Optionally set a “double_opt_in” requirement.
  - If set to “true”, the email will be inserted with a status of “pending”. If “false” the email will be inserted with a status of “subscribed”.

Wunderkind will perform a duplicate check to ensure we are not attempting to create a record for an already-existing subscriber. This is a list-specific check which is done by submitting a request with the email address and list ID and checks to see if an email address is returned.

API Calls for Writing New Subscribers

Request

```
PUT /3.0/lists/{list_id}/members/{md5_hashed_email} HTTP/1.1
Host:{mailchimp_dc}.api.mailchimp.com
Authorization: Bearer {ApiKey-mailchimp_dc}
```

Body - without Interest Categories of Properties

```
{
"email_address":"wkndtest@gmail.com",
"status":"subscribed",
}
```

Body - with Interest Categories of Properties

```
{
{
email_address: “wkndtest@gmail.com”,
status: "subscribed",
merge_fields: {
FNAME: "John",
SOURCE: “wknd”
},
Interests: {
       “Sales & Promos”: true
},
}
```

3. Checking Customer Status

We have several methods for checking a user's customer status in Mailchimp:

1. Checking the number of orders on a member's list-specific profile
2. Checking for a customer on a "store_id" and matching required values
3. Checking for presence on a specific list and/or for required values to match

Method 1:
If a profile is found on the specified list and has more than 0 orders in the "number_of_orders" field, they are considered a customer.

Method 2:
If an ecommerce store has been connected to a Mailchimp account, we can look for a customer by querying their email address on the specific "store_id." If a member is found and has properties matching the required values specified, they will be considered a customer.

Method 3:
Without required values: If a profile is found on the specified list, they will be considered a customer.
With required values: If a profile is found on the specified list AND has properties matching the specified values, they will be considered a customer.

API Calls for Checking Customer Status

Method 1

```
GET /3.0/ecommerce/stores/{storeId}/customers?email_address={email} HTTP/1.1
Host:{mailchimp_dc}.api.mailchimp.com
Authorization: Bearer {ApiKey-mailchimp_dc}
```

Method 2

```
GET /3.0/lists/{list_id}/members/{md5_hashed_email} HTTP/1.1
Host:{mailchimp_dc}.api.mailchimp.com
Authorization: Bearer {ApiKey-mailchimp_dc}
```

Method 3

```
GET /3.0/search-members/?query={query_encoded_eail}&list_id={listId} HTTP/1.1
Host:{mailchimp_dc}.api.mailchimp.com
Authorization: Bearer {ApiKey-mailchimp_dc}
```

4. Checking Mailability

How we check for a user's mailable status in two ways:

1. Checking a member's list-specific "status" property
2. Checking a member's list-specific membership or "merge field" properties

Standard: Status Property Check

If a member profile is found on the specified list:

- If the "status" property has a value of "unsubscribed" we consider them NOT mailable.
- If the "status" property has any other value we consider them mailable.

If a member profile is not found on the specified list:

- Unless told otherwise, we default to an opt-out check and the member will be considered mailable. If an opt-in check is requested, the member will be considered NOT mailable.

Non-Standard: Opt-In Check via List Membership

We can run a custom mailability check based on a member's list membership (eg if a member is ON a list, consider them NOT mailable). There are two ways to considered a contact a list-member:

- Status Property: If a member is found on the list and the "status" property has a value of "subscribed" they are considered a list-member.
- Merge Fields: If a member is found on the list and has Merge Fields matching specified criteria (eg POSTCODE equals 12345) they are considered a list-member.

API Calls for Checking Mailability

Request

```
GET /3.0/lists/{list_id}/members/{md5_hashed_email} HTTP/1.1
Host:{mailchimp_dc}.api.mailchimp.com
Authorization: Bearer {ApiKey-mailchimp_dc}
```

Example Response (shortened)

```
{
    "id": "098ab8765def432ghijkl1234567mnop",
    "email_address": "wkndtest@gmail.com",
    "unique_email_id": "abcdefg1234567",
    "web_id": 12345678,
    "email_type": "html",
    "status": "subscribed",
    "merge_fields": {
        "FNAME": "John",
        "LNAME": "Smith",
        "POSTCODE": "12345",
     },
    "stats": {...
```

5. Updating Unsubscribe Status

The method in which we mark a member as not mailable corresponds to our standard “status” property mailability check:

- We update a member’s list-specific “status” property to “unsubscribed”.
- If a member does not already exist on the specified list, we will create a new record on the list with a status of “unsubscribed”. This is done via the “status_if_new” field in our API request.

Our integration cannot currently update Merge Fields with dynamic data when unsubscribing.

API Calls for Updating Unsubscribe Status

Request

```
PUT /3.0/lists/{list_id}/members/{md5_hashed_email} HTTP/1.1
Host:{mailchimp_dc}.api.mailchimp.com
Authorization: Bearer {ApiKey-mailchimp_dc}
```

Example Response (shortened)

```
{
"email_address":"wkndtest@gmail.com",
"Status_if_new":"unsubscribed",
"status":"unsubscribed"
}
```

6. Sender Options

Wunderkind Sender

With the Wunderkind Sender, we’ll deploy Wunderkind emails through our own ESP. We do not have the ability to send additional data to Mailchimp at time of send.

7. URL Appending

URL Appending is one of Wunderkind's most impactful identification methods. It allows us to identify users returning to your site from marketing messages (non-WKND marketing emails). It is a reverse lookup that we're able to do using a key (unique identifier) that maps to a plaintext email address and appended to all clickout email links.

We use a **Unique Email ID** (the "unique_email_id" property on a member's profile) that is passed to us and link it to its corresponding plaintext email address. Our integration will look to reconcile the Unique Email ID with the value of the "email_address" field.

Within Mailchimp, please make sure the unique email ID value (sometimes referred to as the Mailchimp Subscriber ID, or "mc_eid", within Mailchimp) is appended to all emails.

API Calls for URL Appending

Request 1 - Retrieves the list_id of the campaign sent

```
GET /3.0/campaigns/{campaign_id} HTTP/1.1
Host:{mailchimp_dc}.api.mailchimp.com
Authorization: Bearer {ApiKey-mailchimp_dc}
```

Request 2 - Uses the list_id retrieved along with the unique_email_id to get the list-specific member

```
GET /3.0/lists/{list_id}/members?unique_email_id={unique_email_id} HTTP/1.1
Host:{mailchimp_dc}.api.mailchimp.com
Authorization: Bearer {ApiKey-mailchimp_dc}
```

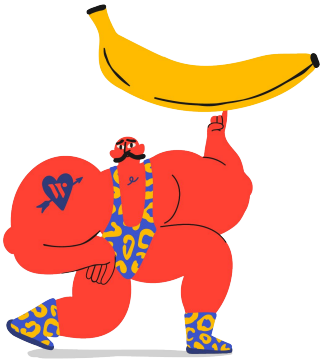
8. Index - Notes, Summary, and Definitions

Important Callouts

- Our integration requires the Mailchimp Data Center value in order to make API requests (see: **Authentication**)
- Our integration requires a **list_id for each request**, as all of our member checks / updates are list-specific.
- We can only update a member's "status" property when unsubscribing (see: **Updating Unsubscribe Status**)
- If a user does not exist in the ESP when unsubscribing, we create a new user within Mailchimp on the specified list. They will be marked as "unsubscribed" on the list and from Wunderkind emails moving forward.

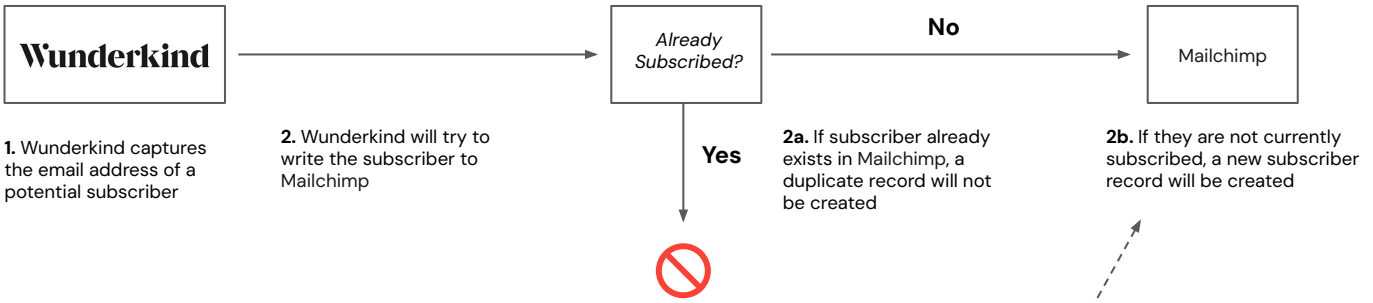
Definitions

- **Authentication:** Verifying the identity of users who want to access an API.
- **Check:** Checking mailability prior to sending a Wunderkind email
    - **Opt-out Check:** If a user is not explicitly marked as not mailable (unsubscribed, not consented) within your ESP, we consider them mailable. Ie. If a user is not in the ESP or does not have a consent event, they will be mailable. This is best practice within the United States.
    - **Opt-in Check:** Only users who are explicitly subscribed or have consent event will be considered mailable. This is best practice outside the United States.
- **Set:** Updating a user's status in the ESP after unsubscribing from a Wunderkind email
- **Wunderkind Sender Integration:** Wunderkind will deploy triggered emails on your behalf through the Wunderkind ESP.
- **Client Sender Integration:** Wunderkind will pass a fully-rendered email to your ESP for deployment, per the Wunderkind Integration Guide.
- **URL Appending:** A reverse lookup within the ESP to identify users that click through a marketing email.



Email Integration Architecture

Onsite Email Capture



Ensure that Wunderkind email submits funnel into the same place we're checking opt-in status.

Triggered Email - Mailability