Wunderkind emarsys

How Wunderkind Integrates With Emarsys

The Emarsys integration with Wunderkind will enable us to collect and send new email addresses to your Emarsys, 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 Emarsys, and more.

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

Table of Contents:

1. Authentication

Emarsys requires a WSSE header to authenticate requests. In order to generate the token for that header and gain access to the Emarsys UI, Wunderkind needs the following information:

- **Requirements:**
  - **API**
    - Username
    - Password
  - **UI Access**
    - Account Name
    - Username
    - Password

2. Writing New Subscribers

Upon email submission, Wunderkind will write new subscribers to Emarsys through the API calls below. Wunderkind may be able to pass additional subscriber information in the body of create and update requests when provided with the API field identifier (number format) and associated value. *(i.e. field identifier: 4 value: first name)*.

Wunderkind will first determine if a submitted email address already exists in Emarsys records with the following API call:

API Calls for Writing New Subscribers

Request
```
POST /api/v2/contact/getdata
Host https://api.emarsys.net
X-WSSE: [wsse-header],
Content-Type: application/json
```

Body
```
{
“key_id”: “3”,
“3”: [email address]
}
```

If there is no existing record with the submitted email address, the following API call will be made to create a new subscriber:

Request
```
POST /api/v2/contact
Host https://api.emarsys.net
X-WSSE: [wsse-header],
Content-Type: application/json
```

Body
```
{
“key_id”: “3”,
“3”: [email address]
}
```

2. Writing New Subscribers (cont.)

If an existing record is found with the submitted email address, Wunderkind can provide 3 separate options for conditionally updating the existing record or rejecting the duplicate submission depending on client preference.

1. **Option 1:** allow all existing email addresses to be resubmitted and update records with any additional field identifiers/values.

```
Request
PUT /api/v2/contact
Host https://api.emarsys.net
X-WSSE: [wsse-header],
Content-Type: application/json

Body
{
“key_id”: “3”,
“3”: [email address]
}
```

2. **Option 2:** allow existing email addresses to be opted in if they have an opted out status in Emarsys and update the record with any additional field identifiers/values. Otherwise, reject as duplicate submission and make no further API requests.

```
Request
POST /api/v2/contact
Host https://api.emarsys.net
X-WSSE: [wsse-header],
Content-Type: application/json

Body
{
“key_id”: “3”,
“external_id”: [email address]
}
```

3. **Option 3:** reject all existing email address submissions regardless of subscriber opt-in status and make no further API requests.

2. Writing New Subscribers (cont.)

After creating/updating the subscriber, an additional API call can be made to **trigger an event**. This is optional and requires the client to provide the desired **event id** and **value.** For example in the API call below, 31 would be the event id and 1 would be the value Wunderkind sets.

Request

```
POST /api/v2/event/[event_id]/trigger
Host https://api.emarsys.net
X-WSSE: [wsse-header], PasswordDigest="[base64(sha1(nonce+date+password))]",
Content-Type: application/json
```

Body

```
{
"key_id": "3",
"3": [email address],
"31": 1
}
```

3. Checking Customer Status

Wunderkind can check a list or segment to determine if a subscriber is a customer who has made a previous purchase or a prospect who has never made a purchase.

1. **Option 1:** List specific customer check
    a. Checks for existence of a subscriber on a specific list. If a subscriber is found on this list, Wunderkind considers the subscriber a previous purchaser. This requires the list id that corresponds with a list of subscribers who have previously made a purchase.

```
Request
GET /api/v2/contactlist/[list_id]/contacts/[contact_id]
Host: api.emarsys.net
X-WSSE: [wsse-header]
Content-Type: application/json
```

2. **Option 2:** Segment specific customer check
    a. Checks for existence of a subscriber on a specific segment. If a subscriber is found on this segment, Wunderkind considers them a customer. Requires the segment id that corresponds with a segment of subscribers who have previously made a purchase.

```
Request
GET/api/v2/filter/[segment_id]/contacts/[contact_id]
Host: api.emarsys.net
X-WSSE: [wsse-header]
Content-Type: application/json
```

4. Checking Mailability

To determine a user's subscription status, Wunderkind can use one of 3 methods.

1. **Option 1: Global subscriber check**
    a. Checks *field identifier* 31. If field 31 has a value of 1 (opted-in), the user is considered mailable. If the field 31 has a value of 2 (opted-out), they are considered unmailable. Subscribers who do not already exist in Emarsys, can all be considered unmailable or mailable depending on preference.

```
Request
POST /api/v2/contact/getdata
Host: api.emarsys.net
X-WSSE: [wsse-header]
Content-Type: application/json

Body
{
"keyId": "3",
"keyValues": [email address],
"fields": ["31"]
}
```

2. **Option 2: List specific subscriber check**
    a. Checks for existence of a subscriber on a specific suppression list. If a subscriber is found on this list, Wunderkind considers them unmailable. Requires client to provide the list id that corresponds with the suppression list.

```
Request
GET /api/v2/contactlist/[list_id]/contacts/[contact_id]
Host: api.emarsys.net
X-WSSE: [wsse-header]
Content-Type: application/json
```

3. **Option 3: Segment specific subscriber check**
    a. Checks for existence of a subscriber on a specific suppression segment. If a subscriber is found on this segment, Wunderkind considers them unmailable. Requires client to provide the segment id that corresponds with the suppression segment.

```
Request
GET/api/v2/filter/[segment_id]/contacts/[contact_id]
Host: api.emarsys.net
X-WSSE: [wsse-header]
Content-Type: application/json
```

5. Updating Unsubscribe Status

The method in which we mark a user as not mailable mirrors how we check a user's mailability globally seen in the Checking Mailability section.

Wunderkind will update *field identifier* 31 to have a value of 2 (opted-out) globally, so that they are considered unmailable.

API Calls for Updating Unsubscribe Status

```
Request
PUT /api/v2/contact/
Host: api.emarsys.net
X-WSSE: [wsse-header]

Body
{
        "keyId": "3",
        "contacts": [
                {
                        "31": "2",
                        "3": [email address]
                }
        ]
}
```

6. Sender Options

Wunderkind is able to deploy an email in two ways: either through Wunderkind's ESP, or through your Emarsys account (this is called "Client Sender").

Wunderkind Sender
With the Wunderkind Sender, we'll deploy Wunderkind emails through our own ESP.

Client Sender
On email send, Wunderkind can create a new subscriber record for subscribers not found in Emarsys global records or consider non-existing records as opted-out and skip sending an email to that subscriber.

```
Request
POST /api/v2/event/[event_id]/trigger
Host: api.emarsys.net
X-WSSE: [wsse-header]
Content-Type: application/json

Body
{
             "key_id": "3",
             “external_id”: [email address],
             “data”:
                          {
                                “html_source”: [email body],
                                “subject”: [email subject line],
                                “from_name”: [client email address],
                          }
             }
```

If writing new subscribers on send:

```
Request
PUT /api/v2/contact/?create_if_not_exists=1
Host: api.emarsys.net
X-WSSE: [wsse-header]
Content-Type: application/json

Body
             {
                   "key_id": "3",
                   “contacts”:
                                [
                                      {
                                            “3”: [email address],
                                      }
                                ]
             }
```

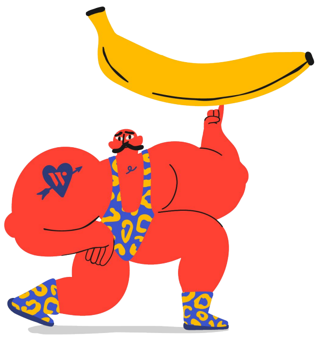
7. Index - Notes, Summary, and Definitions

Important Callouts

- Granting endpoint access via Emarsys for our integration allows seamless connectivity between our systems, enabling smooth data exchange and functionality. With this access, our integration can efficiently leverage Emarsys resources, enhancing operational efficiency and facilitating streamlined processes. Please grant us access to the following endpoints:
  - https://dev.emarsys.com/docs/emarsys-api/ca71341280fb1-create-contacts
  - https://dev.emarsys.com/docs/emarsys-api/1390c2e1ee412-get-contact-data
  - https://dev.emarsys.com/docs/emarsys-api/9200803f3df2a-look-up-contacts-in-a-contact-list
  - https://dev.emarsys.com/docs/emarsys-api/3aeed40c8d82f-get-contact-data-in-a-contact-list
  - https://dev.emarsys.com/docs/emarsys-api/160b46066e62a-count-contacts-in-a-segment
  - https://dev.emarsys.com/docs/emarsys-api/61b720d3c0be6-trigger-an-external-event
  - https://dev.emarsys.com/docs/emarsys-api/b3A6MjQ4OTk4MTE-update-contacts
  - https://dev.emarsys.com/docs/emarsys-api/be7e56b2494c4-look-up-a-contact-in-a-segment
  - https://api.emarsys.net/api/v2/contact/getdata

Definitions

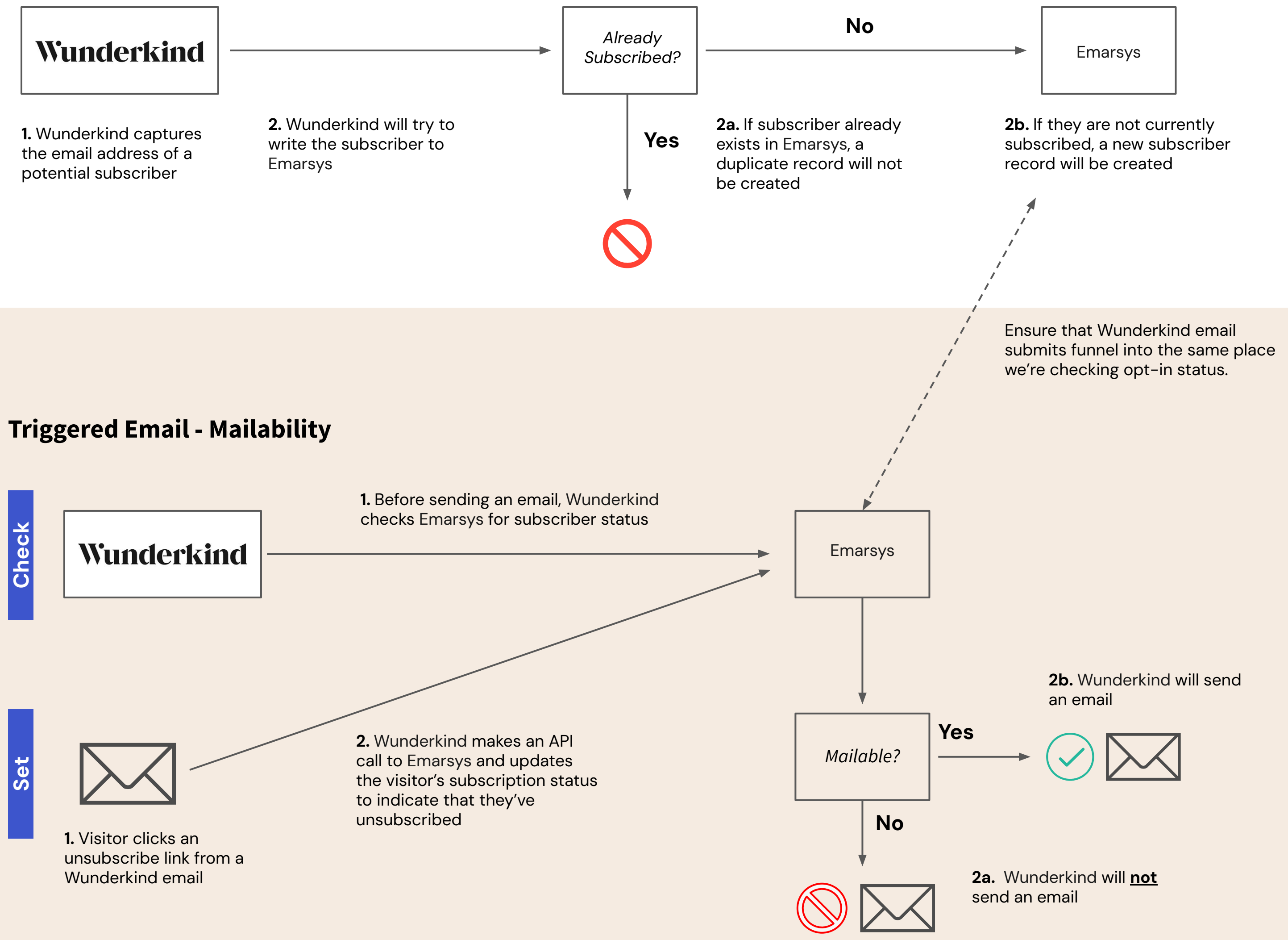
- **Authentication:** Verifying the identity of users who want to access an API.
- **Check:** Checking mailability prior to sending a Wunderkind email
  - **Opt-out Check:** If a user is not explicitly marked as not mailable (unsubscribed, not consented) within your ESP, we consider them mailable. Ie. If a user is not in the Emarsys 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

Triggered Email - Mailability



1. Wunderkind captures the email address of a potential subscriber

2. Wunderkind will try to write the subscriber to Emarsys

2a. If subscriber already exists in Emarsys, a duplicate record will not be created

2b. If they are not currently subscribed, a new subscriber record will be created

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

1. Before sending an email, Wunderkind checks Emarsys for subscriber status

1. Visitor clicks an unsubscribe link from a Wunderkind email

2. Wunderkind makes an API call to Emarsys and updates the visitor's subscription status to indicate that they've unsubscribed

2b. Wunderkind will send an email

2a. Wunderkind will **not** send an email