Open Food Facts API Documentation

1. General Information 2

As a developer, the **Open Food Facts API** allows you to get information about existing products and contribute to the products database.

Using the API, you can create apps to help people make better food choices and also provide data to enhance the database.

Check out how others are making use of the API at https://world.openfoodfacts.org/discover#reuses.

Data Disclaimer

The data contained in the Open Food Facts database are collected by users willing to selflessly contribute to the Open Food Facts initiative.

Therefore, no guarantees can be made for the accuracy, completeness, or reliability of the information provided. The user assumes the entire risk related to the use of data. You (or your users) are very welcome to provide fixes using the **WRITE** API.

Usage

You can use the Open Food Facts API for production use cases, as long as 1 API call equals 1 real scan by a user.

Do you know that we have ready-made SDKs for many programming languages ?

- Cordova: GitHub (old Open Food Facts official app)
- DART: GitHub - Package on pub.dev
- Elixir: GitHub
- Go: GitHub
- NodeJS: GitHub
- PHP: GitHub
- PHP (Laravel): GitHub
- Python: GitHub
- React Native: GitHub
- Ruby: GitHub

Domains

You can either use the global domain (https://world.openfoodfacts.org) or the local domains (https://fr.openfoodfacts.org, https://en.openfoodfacts.org ...) for your API queries.

Endpoint

The Open Food Facts base API endpoint is https://world.openfoodfacts.org/api/2

Version

The current version of the API is 2 .

Authentication

READ and SEARCH operations

No authentication is required.

Add a User-Agent HTTP Header with the name of your app, the version, system and a url (if any), not to be blocked by mistake.

For example: `User-Agent: NameOfYourApp - Android - Version 1.0 - www.yourappwebsite.com`

WRITE operations

No authentication is required for adding new products or adding images.

Basic authentication is required for editing existing products. You can create a global account to let the users of your app contribute without having to create individual credentials in the Open Food Facts site.

Parameters: * `user_id`: YourUserID * `password`: YourPassword

Checking you're authentified

https://world.openfoodfacts.org/cgi/auth.pl

This endpoint returns status 200 or 403 if the user is authentified, either through the "session" cookie, or with the user_id and password parameters.

Environments

You can do READ / SEARCH operations on the `prod` environment running @ https://world.openfoodfacts.org.

You can do WRITE operations tests on the `dev` environment running @ https://world.openfoodfacts.net (user:`off`, password:`off`).

Security

Use the SSL version of the API: https://world.openfoodfacts.org

---

Error Codes

- **Product does not exist** - HTTP code 200 + "status_verbose" : "product not found" + "Status" : 0. The request format is correct, but the product code does not exist in the database.
- **Wrong Password** - HTTP code 200 + an HTML page with a link to log in. The request format is correct, but basic authentication is missing or the password entered is not correct.
- **Server down** - HTTP codes 502/503/500
- **Redirect to another product** - HTTP code 301

Disclaimer: The HTML code 404 is never thrown, even when a wrong password is entered. A feature request has been created and we are already working to fix this.

Rate limit

The API intended use is for apps, with one real user scan = one query. Automated queries are not supported. Please let us know in advance if you expect a high volume of calls.

For more information, see: https://world.openfoodfacts.org/data

---

Cache

Some queries (facets) are caches. Should you need to disable the cache, you can pass the `nocache=1` parameter.

Payload size reduction

Using the `fields=` parameter, you can reduce the response to only the fields you need.

Preliminary Considerations

The API development is in progress. This has several implications:

- Open Food Facts and food products are constantly evolving.
- Assume that data is less reliable until the product is marked as complete. You might want to filter incomplete products to avoid issues (this is especially relevant for allergens or food intolerances). Let your end users know about this and encourage them to exercise caution. Be upfront about possible risks. You can use the following template to inform your users: *The data provided to you by this app are retrieved from the Open Food Facts database. No guarantees can be made for the accuracy, completeness, or reliability of the information provided. The data are provided "as is" and the originating source for the data (Open Food Facts) is not liable for any damages arising out of the use of the data.*
- Join our Slack Channel (https://slack-ssl-openfoodfacts.herokuapp.com) to get help, to share your thoughts, to let us know what you build with the API (contact@openfoodfacts.org or in the #API channel) or if you want to use WRITE operations.
- You can also join the mailing list to be notified when improvements or changes are made to the API (we send only relevant information and very few e-mails. Don't worry, you won't be spammed). To join the mailing list, send an empty e-mail to api-subscribe@openfoodfacts.org to subscribe.

License

- Do not send copyrighted photos or information using the API. Everything you send is OdBL for the data (https://opendatacommons.org/licenses/odbl/summary/index.html) and CC-BY-SA for the pictures (https://creativecommons.org/licenses/by-sa/4.0/). If you don't own the data, you bear all the legal consequences.
- Mention Open Food Facts as the source of the data.
- Do not mix with other product databases (since you are then required to release them under OdBL, at your own legal risk).
- Share any additions under the OdBL with the community.
- By using any part of the API you have read and understood the license.

API Conventions

- Fields that end with `_t` are dates in the UNIX timestamp format (number of seconds since Jan 1st 1970)
- Fields that end with `_datetime` are dates in the ISO8601 format: yyyy-mm-ddThh:mn:ssZ
- Fields that end with `_tags` are comma-separated list of tags (e.g. categories_tags is the set of normalized tags computer from the categories field)
- Fields that end with a language 2 letter code (e.g. fr for French) is the set of tags in that language
- Fields that end with `_100g` correspond to the amount of a nutriment (in g) for 100 g or 100 ml of product

---

Bugs

Do not hesitate to file a bug if you find an issue in the API or need an improvement. You can fill out the issue report on GitHub:

- General bugs: https://github.com/openfoodfacts/openfoodfacts-server/issues
- API bugs: https://github.com/openfoodfacts/openfoodfacts-server/labels/api
- API milestone: https://github.com/openfoodfacts/openfoodfacts-server/milestone/8

Downloading Data

It is recommended to use the live API to get updated data about products. However, in some cases, you may need a snapshot. They are available at:

- https://world.openfoodfacts.org/data (all data)
- https://[countrycode].openfoodfacts.org/data (data for a specific country).

Example: https://us.openfoodfacts.org/data - (See the list of countries in the **Countries** taxonomy)

Exporting Data

- File Encoding: The file encoding is Unicode UTF-8.
- CSV API: The character that separates fields is < tab > (tabulation).
- JSON

API Roadmap

API Redesign: The API is far from perfect. It's been decided to fix the most urgent bugs and start planning for a new version, more compliant with modern API standards. We need all the help we can get. Please join us on the #api Slack channel.

- Project API: Additives
- Project API: States
- Project API: Statistics
- Project API: Statistics Entry Dates

Other Projects

- Open Pet Food Facts
- Open Beauty Facts
- Open Products Facts

More topics

- See **5. Filtering** section for a list of all available API parameters.
- See **6. Understanding responses** to figure out the response data fields.

1.2 Authentication | _

Get all food products | GET _ https://world.openfoodfacts.org

Description

Get all products from Open Food Facts API.

Query

Key	Value	Description
json	true	

Select Example Request/Response

2. READ requests 2

READ requests allow you to retrieve the nutritional data of a product with a barcode.

Get nutrition facts for a specific barcode | GET _ https://world.openfoodfacts.org/api/v2/product/04963406

Description

- Add `product/<BARCODE>` to locate the product by it's barcode.

Select Example Request/Response

Get a country-specific nutrients ordered list | GET _ https://us.openfoodfacts.org/cgi/nutrients.pl

Description

Get a country-specific nutrients ordered list. It changes based on country and is useful both to show a nutrition table or a nutrition input form.

https://fr.openfoodfacts.org/cgi/nutrients.pl

https://us.openfoodfacts.org/cgi/nutrients.pl

3. SEARCH Requests 6

SEARCH requests allow you to retrieve the nutritional data of products that comply with your search criteria. Check out the examples below to see what you can do !

Important! The search feature works on whole words only, not parts of words. Your application **should not** have "search as you type" features that send search queries with parts of words, since this causes performance issues on the Open Food Facts server.

Search for Spanish products | GET _ https://es.openfoodfacts.org/cgi/search.pl

Description

- Add `es.` prefix to get only Spanish products.
- Add `json=true` to get a JSON response.

Headers

Key	Value	Description
Content-Type	application/json	

Query

Key	Value	Description
action	process	
json	true	

Search for US breakfast cereals | GET _ https://us.openfoodfacts.org/cgi/search.pl

Description

- Add `us.` prefix to get only US products.
- Add `json=true` to get a JSON response.
- Add a `categories` filter to get only breakfast cereals.

Query

Key	Value	Description
action	process	
tagtype_0	categories	
tag_contains_0	contains	
tag_0	breakfast_cereals	
json	true	

Search for Italian foods with a great Nutriscore (A) | GET https://it.openfoodfacts.org/cgi/search.pl

Description

- Add `it.` prefix to get only Italian products.
- Add `json=true` to get a JSON response.
- Add a `nutrition_grade` filter to get food with Nutriscore 'A'

Query

Key	Value	Description
action	process	
tagtype_1	nutrition_grades	
tag_contains_1	contains	
tag_1	A	
json	true	

Search for French breakfast cereals with no additives nor palm oil and a great Nutriscore (A) | GET https://fr.openfoodfacts.org/cgi/search.pl

Description

- Add `fr.` prefix to get only French products.
- Add `json=true` to get a JSON response.
- Add multiple criteria (AND):
  - Add a `categories` filter to get only breakfast cereals
  - Add a `nutrition_grade` filter to get food with Nutriscore 'A'
  - Add `ingredients_from_palm_oil=without` to get food without palm oil
  - Add `additives=without` to get food without additives

Query

Key	Value	Description
action	process	
tagtype_0	categories	
tag_contains_0	contains	
tag_0	breakfast_cereals	
tagtype_1	nutrition_grades	
tag_contains_1	contains	
tag_1	A	
ingredients_from_palm_oil	without	
additives	without	
json	true	

Select Example Request/Response

Get a list of products by barcodes | GET https://world.openfoodfacts.org/api/v2/search

Description

https://world.openfoodfacts.org/api/v2/search?code=8024884500403,3263855093192&fields=code,product_name

This API is limited by the largest header default limit of Nginx (8K). If you're requesting EAN13 barcodes the server should allow you $^{8000}/_{14}$=571 products (I put 14 because you need to add a comma between each EAN).

For cross platform sharing, you can also build a link to the web version of Open Food Facts: https://world.openfoodfacts.org/search?code=8024884500403,3263855093192

Query

Key	Value	Description
code	8024884500403,3263855093192	List of the barcodes you want to get values for
fields	code,product_name	Optional, to reduce payload size to just what you need

Get suggestions to help in adding/editing a product | GET https://world.openfoodfacts.org/cgi/suggest.pl

Description

https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=emb_codes&term=FR https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=categories&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=labels&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=ingredients&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=packaging_shapes&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=packaging_materials&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=packaging_shapes&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=languages&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=stores&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=brands&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=countries&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=traces&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=origins&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=states&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=nutrients&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=additives&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=allergens&term=f https://world.openfoodfacts.org/cgi/suggest.pl?tagtype=minerals&term=f

Query

Key	Value	Description
tagtype	emb_codes	The kind of values you want. Can be: emb_codes, categories, labels, ingredients, packaging_shapes, packaging_materials, languages, stores, countries, traces, states, origins, nutrients, additives, allergens, minerals
term	FR	The value you'd like to get suggestions for

4. WRITE Requests 7

WRITE requests allow you to contribute new products and data to the Open Food Facts database.

Note: Please use the `dev` environment `https://world.openfoodfacts.net` for making test write calls (user:`off`, password:`off`). Remember to join the API channel on Slack before making a POST request !

Note:

Selecting, cropping and rotating photos are non-destructive actions. That means, the original version of the image uploaded to the system is kept as is. The subsequent changes made to the image are also stored as versions of the original image.

The actions described in this topic do not modify the image, but provide metadata on how to use it (the data of the corners in the case of selection and the data of the rotation). That is, you send an image to the API, provide an id, you define, for example, the cropping and rotation parameters and as a response, the server generates a new image as requested and you can call this new version of the image.

Add a new product | POST https://us.openfoodfacts.org/cgi/product_jqm2.pl

Query

Key	Value	Description
code	04963406	
user_id	test	
password	test	
brands	Häagen-Dazs	
labels	kosher	

Select Example Request/Response

Add a photo to an existing product | POST https://us.openfoodfacts.org/cgi/product_jqm2.pl

Description

Photos are source and proof of data. When you upload an image to Open Food Facts, the image is stored as is. The first photo uploaded for a product is auto-selected as the product's "front" photo.

Before uploading photos:

- **Image Quality:** Uploading quality photos of a product, its ingredients and nutrition table is very important, since it allows the Open Food Facts OCR system to retrieve important data to analyze the product. The minimal allowed size for photos is 640 x 160 px.
- **Upload Behavior:** In case you upload more than one photo of the front, the ingredients and the nutrition facts, beware that only the first photo of each category will be displayed. (You might want to take additional images of labels, recycling instructions, and so on). All photos will be saved.
- **Label Languages:** Multilingual products have several photos based on languages present on the packaging. You can specify the language by adding a lang code suffix to the request.

Product Image Upload (Perl):

The API route is `product_image_upload.pl` and you can specify from which perspective the photo was taken, by sending the imagefield to precise the angle, AND the image as a Multipart response in the matching field.

Parameters:

- `code`: the barcode of the product
- `imagefield`: (can be either: front | ingredients | nutrition | packaging)
- `imgupload_front` : your image file if imagefield=front
- `imgupload_ingredients`: your image file if imagefield=ingredients
- `imgupload_nutrition`: your image file if imagefield=nutrition
- `imgupload_packaging`: your image file if imagefield=packaging

Query

Key	Value	Description
code	04963406	
product_image_upload.pl/imgupload_front	cheeriosfrontphoto.jpg	

Select Example Request/Response

Crop a photo | POST https://world.openfoodfacts.org/cgi/product_image_crop.pl

Description

Note: Cropping is only relevant for editing existing products. You cannot crop an image the first time you upload it to the system.

Query

Key	Value	Description
code	04963406	
imgid	2	
id	front_en	
x1	0	
y1	0	
x2	145	
y2	145	

Select Example Request/Response

Rotate a photo | POST https://world.openfoodfacts.org/cgi/product_image_crop.pl

Description

Although we recommend rotating photos manually and uploading a new version of the image, the OFF API allows you make api calls to automate this process.

You can rotate existing photos by setting `angle` to 90^0, 180^0 or 270^0 clockwise.

Example:

POST `https://world.openfoodfacts.org/cgi/product_image_crop.pl?code=3266110700910&id=nutrition_fr&imgid=1&angle=90`

Query

Key	Value	Description

Key	Value	Description
code	04963406	
id	nutrition_fr	
imgid	1	
angle	90	

Select Example Request/Response

Deselecting a photo | POST

Description

You have to deselect photos to remove languages that are not relevant to the product.

[DOCUMENTATION TBA]

Select Example Request/Response

Performing OCR on a product | GET https://world.openfoodfacts.org/cgi/ingredients.pl

Description

Open Food Facts uses optical character recognition (OCR) to retrieve nutritional data and other information from the product labels.

Process

1. Capture the barcode of the product where you want to perform the OCR.
2. The Product Opener server software opens the image (`process_image=1`)
3. Product Opener returns a JSON response. Processing is done using Tesseract or Google Cloud Vision (recommended). The result is often cripped with errors with Tesseract, less with Google Cloud Vision.

Notes: * The OCR may contain errors. Encourage your users to correct the output using the ingredients WRITE API. * You can also use your own OCR, especially if to plan to send a high number of queries.

OCR with Google Cloud Vision

We recommend Google's Vision API to detect and extract text from the images.

For more information about this product, see: https://cloud.google.com/vision/docs/ocr?hl=en

Set `ocr_engine=google_cloud_vision` to use it.

Query

Key	Value	Description
code	04963406	
id	ingredients_fr	You can also pass packaging_fr if you want to extract recycling instructions.
process_image	1	
ocr_engine	google_cloud_vision	

Select Example Request/Response

Image Refresh API | GET https://fr.openfoodfacts.org/api/v0/produit/3483130043180/beurre-cru-a-la-baratte-les-petites-laiteries

Description

The image refresh API is to ensure we can request photo updates for select products to users for images which should be taken or re-taken (because they are too old, or possibly too small / blurry).

https://fr.openfoodfacts.org/api/v0/produit/3483130043180/beurre-cru-a-la-baratte-les-petites-laiteries?fields=images_to_update_fr

It returns a hash of image types + language code (only for the requested language code which should be the language of the app). The value is 0 for images we don't have, or the age of the image (in seconds) for apps that want to add some context like `"Our photo of the ingredients is 14 months old, could you take a new one?"`.

Sample API response

```
product: {
images_to_update_fr: {
packaging_fr: 0,
front_fr: 83734290,
ingredients_fr: 83734290
}
},
```

- [ ] Possible languages (already loaded in the app) https://static.openfoodfacts.org/data/taxonomies/languages.json
- [ ] Possible fields (front, ingredients, nutrition, packaging)

Pseudo code to generate button text

for field in images_to_update_fr: if field.value=0 verb = "take" else: verb = "refresh" field*name = field.split.before("")* field*language = field.split.after("")* button_text = fetch_button_text(field_name, field_language, verb)

Strings to combine (suffix the language at the end)

"Take %s picture" "Refresh %s picture" "ingredients" "front" "nutrition" "packaging"

Optional - mention how old the image is

How to convert seconds in human readable format: 83734290 = 2 years and 7 months (example routine to convert) - https://stackoverflow.com/questions/29681328/convert-seconds-into-years-months-weeks-hours-minutes-and-seconds

Send the right query based on the initial field in images_to_update_fr

- Do not use the computed values in the pseudo code
- Probably refactor the methods we currently have to pass the field name, and make it future proof if we want to add new image fields.

Query

Key	Value	Description
fields	images_to_update_fr	

5. Filtering 3

Advanced filtering is available to make fine-grained requests to the API.

General parameters | GET

Description

This section includes the parameters you can add to make `READ` / `SEARCH` / `WRITE` requests.

URL Parameters

Country code

A **country code** is prefixed to the domain name `openfoodfacts.org`, e.g: `fr.`: it allows filtering all products from a specific country.

You can use `world.` to display products from all over the world or use one of the Alpha 2 codes as per ISO-3166-1.

Examples: - United States: `us` - France: `fr` - Spain: `es`

You can find the full list of supported country codes in the **Countries** taxonomy.

Important! Using a specific country code will also change the naming of the response fields, see language code.

Language code

A **language code** can be added after the **country code** to specify the language of the response fields, e.g: `https://<cc>-<lc>.openfoodfacts.org`

Example:

https://fr.openfoodfacts.org/categorie/pizzas.json

- Products returned are sold in France.
- Names of the response fields are in French.

Example 2:

https://fr-en.openfoodfacts.org/category/pizzas.json

- Products returned are sold in France.
- Names of the response fields are in English.

The language codes supported are based on the **ISO Standards 639-1**.

You can find the full list of supported language codes in the **Languages** taxonomy.

API Version

Current version number of the Open Food Facts API. For now, only version 0 is available. To be represented as: `/api/v0`

Results per page

`page_size` # `page_size`

- `20` # 20
- `50` # 50
- `100` # 100
- `250` # 250
- `500` # 500
- `1000` # 1000

Pagination

- `page=1`

Format

- `json=true` (recommended)
- `xml=true`

Output fields

Filtering the output fields reduces the payload size, the bandwith needed and the download time. To filter the fields, simply add the “fields“ parameter to your search query. Example to retrieve only the `generic_name`: https://world.openpetfoodfacts.org/api/v0/product/20106836.json?fields=generic_name

---

SEARCH parameters | GET

Description

This section includes the parameters you can add to make `SEARCH` requests.

URL Parameters

You can use any of the fields used on the website search form.

There are three types of parameters you can use to filter the results:

- **Criteria**
- **Ingredients**
- **Nutriments**

Criteria

Every time you use a criterion in your query, you must use the following tags:

- `tagtype_0=categories`
- `tag_contains_0=contains`
- `tag_0=cereals`

Example: https://world.openfoodfacts.org/cgi/search.pl?action=process&tagtype_0=categories&tag_contains_0=contains&tag_0=cereals

Where `tagtype_0` can be one of the following:

- `brands`
- `categories`
- `packaging`
- `labels`
- `origins`
- `manufacturing_places`
- `emb_codes`
- `purchase_places`
- `stores`
- `countries`
- `additives`
- `allergens`
- `traces`
- `nutrition_grades`
- `states`
- `contains`
- `does_not_contain`

If you want to add more criteria to the query, increase the number of the tag. For example:

- `tagtype_0=categories`
- `tag_contains_0=contains`
- `tag_0=cereals`
- `tagtype_1=label`
- `tag_contains_1=contains`
- `tag_1=kosher`

Ingredients

Use the following parameters to include or exclude products containing any additives or ingredients from palm oil:

- additives
  - `without_additives`
  - `with_additives`
  - `indifferent_additives`
- ingredients_from_palm_oil
  - `without`
  - `with`
  - `indifferent`
- ingredients_that_may_be_from_palm_oil
  - `without`
  - `with`
  - `indifferent`
- ingredients_from_or_that_may_be_from_palm_oil
  - `without`
  - `with`
  - `indifferent`

Nutriments

You can also filter by nutriments (fat, sugars, energy, etc). To do so, you need to add three different parameters for each nutriment:

Example

- `nutriment_0=energy`
- `nutriment_compare_0=lt`
- `nutriment_value_0=500`

You can enter the following categories (nutriment_0): https://static.openfoodfacts.org/data/taxonomies/nutrients.json

Comparison of nutriments

Nutriment to compare

`nutriment_compare_0`

Operator

- `lt` # less than
- `lte` # less than or equal
- `gt` # greater than
- `gte` # greater than or equal
- `eq` # equal to
- `nutriment_value_0` - Value to compare the nutrients to

Other search parameters

Output

- `sort_by` # sort by
- `unique_scans_n` # Popularity
- `product_name` # Product name
- `created_t` # Add date
- `last_modified_t` # Edit date

Linked Data

Whenever possible, Open Food Facts entities are linked to Wikidata, and in turn to Wikipedia. What this means is that you get access to a trove of additional encyclopedic knowledge about food. You can for instance get: Wikipedia articles about Camembert, the translation of salt in many languages, the molecular structure of a cosmetic ingredient... We provide the Wikidata QID, which is an unambiguous, stable and reliable identifier for a concept that will be useful to actually retrieve info from Wikipedia and Wikidata.

Example

https://world.openfoodfacts.org/categories.json

```
{"linkeddata":{"wikidata:en":"Q40050"},"url":"https://world.openfoodfacts.org/category/beverages","name":"Beverages","id":"en:beverages","products":14196}
Beverages >> https://world.openfoodfacts.org/category/beverages >> Q40050 >> https://www.wikidata.org/wiki/Q40050
```

WRITE parameters | GET

Description

This section includes the parameters you can add to make `WRITE` requests.

URL Parameters

`user_id` and **`password`**

No authentication is required for adding new products or adding images, although it is recommended.

Basic authentication is required for **editing existing products**.

You can create a global account to let the users of your app contribute without having to create individual credentials in the Open Food Facts site.

`code`

The word `code`, followed by the product barcode must be added to the URL:

https://us.openfoodfacts.org/cgi/product_jqm2.pl?code=0074570036004

Additional field values (new product)

You can add several values to a field by adding a comma between them.

Example: `labels="labelA, labelB"`

Reading back, use `labels_tags` to get an array of labels.

Additional field values (existing product)

To add additional information to an existing product field, add the prefix `add_` to the parameter name.

```
POST https://us.openfoodfacts.org/cgi/product_jqm2.pl?code=0074570036004&user_id=myappname&password=******&add_categories=Desserts
```

Example

```
https://world.openfoodfacts.org/cgi/product_jqm2.pl?code=0048151623426&user_id=usernameexample&password=*****&product_name=Maryland%20Choc%20Chip&quantity=230g&brands=Golden%20Cookies&nutriment_energy=450&nutriment_energy_unit=kJ&nutrition_data_per=serving&
ingredients_text=Fortified%20wheat%20flour%2C%20Chocolate%20chips%20%2825%25%29%2C%20Sugar%2C%20Palm%20oil%2C%20Golden%20syrup%2C%20Whey%20and%20whey%20derivatives%20%28Milk%29%2C%20Raising%20agents%2C%20Salt%2C%20Flavouring&traces=Milk%2C+Soya%2C+Nuts%2C+Wheat
```

Note: Use `%20` for spaces (e.g. Maryland%20Choc%20Chip), `&` to concatenate parameters (e.g. quantity=230g&brands=Golden%20Cookies) and `=` to link the parameter to the value (e.g. nutriment_energy=450).

Breakdown:

- **Server url + barcode**: `https://world.openfoodfacts.org/cgi/product_jqm2.pl?code=0048151623426`
- **User id**: `usernameexample`
- **Password**: `password`
- **Product name**: `product_name=Maryland%20Choc%20Chip`. **Important:** German umlauts are not converted (e.g. ä -> ae). For more information, see the **FAQ** section.
- **Quantity**: `quantity=230g`
- **Brands**: `brands=Golden%20Cookies`
- **Energy**: `nutriment_energy=450`
- **Nutrition data per**: `nutrition_data_per=serving`
- **Ingredients**: `Fortified%20wheat%20flour%2C%20Chocolate%20chips%20%2825%25%29%2C%20Sugar%2C%20Palm%20oil%2C%20Golden%20syrup%2C%20Whey%20and%20whey%20derivatives%20%28Milk%29%2C%20Raising%20agents%2C%20Salt%2C%20Flavouring&traces=Milk%2C+Soya%2C+Nuts%2C+Wheat`

Other Parameters:

- `Nutriment_energy_unit`: possible values are: kj, kcal. This value always applies to the nutriment_energy value. The normalized energy value, in kJ, can be found in `energy_100g`.

Status Codes

- Valid edits get the following response: `{ ... "status_verbose": "fields saved", "status": 1 ... }`
- If the password entered is not correct, an HTML 200 code + an HTML page with a link to login is displayed.
- If the code is not correct, you get a 0 response.

About Copyright

Make sure you don't upload photos or information with copyright. Everything you send is OdBL for the data, and CC-BY-SA for the pictures. Be aware that you bear the legal consequences for uploading protected content.

6. Understanding responses 5

Product | GET

Description

General information:

- `code` : barcode of the product (can be EAN-13 or internal codes for some food stores). For products without a barcode, Open Food Facts assigns a number starting with the 200 reserved prefix.
- `url` : url of the product page on Open Food Facts.
- `creator` : contributor who first added the product.
- `created_t` : date when the product was added (UNIX timestamp format).
- `created_datetime` : date when the product was added (ISO8601 format: yyyy-mm-ddThh:mn:ssZ).
- `last_modified_t` : date when the product page was last modified.
- `last_modified_datetime`: date and time when the product was last modified.
- `product_name` : name of the product.
- `generic_name`: legal name of the product as regulated by the European authorities.
- `quantity` : quantity and unit.

Ingredients:

- `ingredients_text`: Raw list of ingredients. This will get automatically parsed and get used to compute the Eco-Score. You can either request if (if it exists) or send it in a specific language, e.g: `ingredients_text_en`
- `traces`: List of substances that might cause allergies that are present in trace amount in the product (this does not include the ingredients, as they are not only present in trace amount). It is taxonomized with the allergens taxonomy.
- `traces_tags`

Packages:

- `packaging` : shape, material. Example: Cardboard
- `packaging_tags`
- `packaging_text` : Recycling instructions as raw text eg: "Plastic bottle to recycle, Plastic cap to recycle". This will get automatically parsed and get used to compute the Eco-Score. You can either request if (if it exists) or send it in a specific language, e.g: `packaging_text_en` for the above example
- `emb_codes` : packager code. Example: EMB 2013330
- `emb_codes_tags`

Brands:

- `brands`
- `brands_tags`

Categories:

- `categories`
- `categories_tags`

Location:

- `origins` : origins of ingredients
- `origins_tags`
- `first_packaging_code_geo` : coordinates corresponding to the first packaging code.
- `manufacturing_places` : places where the product was manufactured or transformed.
- `manufacturing_places_tags`
- `cities`
- `cities_tags`
- `purchase_places`: country, state and/or city where the product can be purchased. For example: Paris, France.
- `stores`: distributor name. Example: Tesco, Walmart, Carrefour.
- `countries` : list of countries where the product is sold.
- `countries_tags`

Labels

- `labels`: Example: vegan, fat free, Kosher.
- `labels_tags`

Producer

- `producer`
- `producer_product_id`
- `producer_version_id`

Value and Weight

- `net_weight_value`
- `net_weight_unit`
- `drained_weight_value`
- `drained_weight_unit`
- `volume_value`
- `volume_unit`

Images

- `image_url`
- `image_small_url`: simplified version of the url.

Energy

- Legacy
  - `energy_unit`: (string). The unit used in the `energy_value` field (example in JSON: "energy_unit":"kJ"). Possible values are "kJ" or "kcal".
  - `energy_value`: (string). The standardized value of a serving of 100g (or 100ml for liquids) for energy expressed in the unit specified in the field `energy_unit` (example in JSON: "energy_value":"190").
- Preferred method
  - `energy-kj_unit`: (string). The unit used in the field energy-kj_value (example in JSON: "energy_unit":"kJ"). The only possible value is "kJ";
  - `energy-kj_value`: (string). The standardized value of a serving of 100g (or 100ml for liquids) for energy expressed in kJ (example in JSON: "energy-kj_value":"190").
  - `energy-kcal_unit`: (string). The unit used in the field energy-kcal_unit (example in JSON: "energy_unit":"kcal"). The only possible value is "kcal";
  - `energy-kcal_value`: (string). The standardized value of a serving of 100g (or 100ml for liquids) for energy expressed in kcal (example in JSON: "energy-kcal_value":"190").

According to the European regulation, the ratio between values calculated in kJ and values calculated in kcal may differ from the standard conversion ratio of 4.184 (because of carried rounding errors). Both values might appear on the same product. In that case, the value in kJ will be the one returned in the legacy `energy_unit` and `energy_value` fields. If only one unit was provided (kJ or kcal), this unit will be returned in the legacy `energy_unit` and `energy_value` fields.

Additives:

- `additives_n` : number of food additives
- `additives`
- `additives_tags`

Miscellaneous:

- `serving_size` : serving size in g (or ml)
- `serving_quantity`
- `no_nutriments` : indicates if the nutrition facts are shown on the product label.

- `ingredients_text`
- `allergens`
- `traces`
- `ingredients_from_palm_oil_n`
- `ingredients_from_palm_oil`
- `ingredients_from_palm_oil_tags`
- `ingredients_that_may_be_from_palm_oil_n`
- `ingredients_that_may_be_from_palm_oil`
- `ingredients_that_may_be_from_palm_oil_tags`
- `nutrition_grade_fr` : nutrition grade ('a' to 'e'). see https://world.openfoodfacts.org/nutriscore
- `main_category`
- `other_information`
- `conservation_conditions`: Example: Keep in a dry place.
- `recycling_instructions_to_recycle`
- `recycling_instructions_to_discard`
- `nutrition_grade_fr_producer`: declarative (printed on the packaging)
- `recipe_idea`
- `customer_service`: contact info of customer service.
- `preparation`: how to cook the food: microwave, oven, which temperature...
- `warning`: regulatory warning. Example: contains sorbitol.
- `data_sources`: source of data imported from producers.
- `nova_group`: system of grades for comparing the degree of processing of products. For more information, see: https://world.openfoodfacts.org/nova
- `pnns_groups_1`: disregard. Used to improve the nutriscore calculation.
- `pnns_groups_2`: disregard. Used to improve the nutriscore calculation.
- `states`: if the product is complete or if there is any information missing.

Environment
The Eco-Score needs to be queried according to the country of the user.
Due to the recent nature of the Eco-Score, the full APIs are documented in a separate document.
https://docs.google.com/document/d/1_5AeofpXbaKY9Rd3eeWmHIrhJE8GiPQ-Mfx1SCvpzME/edit?usp=sharing

Other nutrition keys

- `carbon-footprint_100g` : carbon footprint (indicated on some products). The unit is absolute grams of C02.
- `ph_100g` : pH (no unit)
- `cocoa` : minimal cacao content of the product in %. **Important!**: Note the typo.
- `fruits-vegetables-nuts_100g` : % of fruits, vegetables and nuts (excluding potatoes, yams, manioc)
- `nutrition-score-fr_100g` : experimental nutrition score derived from the UK FSA score and adapted for the French market (formula defined by the team of Professor Hercberg)
- `nutrition-score-uk_100g` : nutrition score defined by the UK Food Standards Administration (FSA).
  For more information about the difference between the `fr` and `uk` nutri-scores see the **FAQ** section of this documentation.

Nutrition facts | GET

Description

Each nutrition fact consists of multiple fields which are represented by a key. The fields can also be found in the taxonomy translation file.

Field names are built by concatenating 3 concepts:

- **The nutriment**: "fiber", "carbohydrate", "salt", etc...
- **as sold vs prepared**: "" (nothing is added for as sold) or "prepared" (for prepared products. Example: dehydrated soups, instant cocoa or convenience products like fries).
- **The reference quantity**: "100g" or "serving"

Example 1: 3.4 g of carbohydrates in the product as sold for 100g should be represented as:

`carbohydrates_100g`: 3.4

Example 2: 12 mg of zinc in the prepared product for a serving of 125 mL should be defined as:

`zinc_prepared_serving`: 0.012

Important: * Only the nutrition facts that are actually found on the packaging are present in the interface. * `key_serving` and `Key_100g` are values for the serving size or 100g. One of them is equal to "key", the other one is converted.

Main nutrition keys:

- `energy`
- `proteins`
- `casein`
- `serum-proteins`
- `nucleotides`
- `carbohydrates`
- `sugars`
- `sucrose`
- `glucose`
- `fructose`
- `lactose`
- `maltose`
- `maltodextrins`
- `starch`
- `polyols`
- `fat`
- `saturated-fat`
- `butyric-acid`
- `caproic-acid`
- `caprylic-acid`
- `capric-acid`
- `lauric-acid`
- `myristic-acid`
- `palmitic-acid`
- `stearic-acid`
- `arachidic-acid`
- `behenic-acid`
- `lignoceric-acid`
- `cerotic-acid`
- `montanic-acid`
- `melissic-acid`
- `monounsaturated-fat`
- `polyunsaturated-fat`
- `omega-3-fat`
- `alpha-linolenic-acid`
- `eicosapentaenoic-acid`
- `docosahexaenoic-acid`
- `omega-6-fat`
- `linoleic-acid`
- `arachidonic-acid`
- `gamma-linolenic-acid`
- `dihomo-gamma-linolenic-acid`
- `omega-9-fat`
- `oleic-acid`
- `elaidic-acid`
- `gondoic-acid`
- `mead-acid`
- `erucic-acid`
- `nervonic-acid`
- `trans-fat`
- `cholesterol`
- `fiber`
- `sodium`
- `alcohol`: % vol of alcohol
- `vitamin-a`
- `vitamin-d`
- `vitamin-e`
- `vitamin-k`
- `vitamin-c`
- `vitamin-b1`
- `vitamin-b2`
- `vitamin-pp`
- `vitamin-b6`
- `vitamin-b9`

- vitamin-b12
- biotin
- pantothenic-acid
- silica
- bicarbonate
- potassium
- chloride
- calcium
- phosphorus
- iron
- magnesium
- zinc
- copper
- manganese
- fluoride
- selenium
- chromium
- molybdenum
- iodine
- caffeine
- taurine

Tags | GET

Description

The `tags` suffix you find in some of the response fields refer to the normalized version of the values using the taxonomies. If a value is not taxonomized, it will be displayed in the original language. Example: `es:leche` (original language, original text). Note that not all special characters are supported. For more information, see the **FAQ** section.

Attributes API | GET https://fr.openfoodfacts.org/api/v0/produit/3700214614266.json

Description

The Attributes API is aimed at simplifying personal search and personalization of results for apps. It will be documented here once it's ready.

Sample output:

https://fr.openfoodfacts.dev/api/v0/produit/3700214614266/chocolat-noir-perou-90-fruite-et-boise-alter-eco?fields=product_name,code,attributes_en

{ product: { product_name: "Chocolat noir Pérou 90% fruité et boisé", code: "3700214614266", attributes_en: [{ id: "labels", name: "Labels", attributes: [{ status: "known", id: "labels_organic", description_short: "Promotes ecological sustainability and biodiversity.", title: "Organic product", description: "Organic farming aims to protect the environment and to conserve biodiversity by prohibiting or limiting the use of synthetic fertilizers, pesticides and food additives.", name: "Organic product", match: 100 }, { status: "known", description_short: "Fair trade products help producers in developping countries.", id: "labels_fair_trade", title: "Fair trade product", description: "When you buy fair trade products, producers in developing countries are paid an higher and fairer price, which helps them improve and sustain higher social and often environmental standards.", name: "Fair trade product", match: 100 }] }] }, code: "3700214614266", status: 1, status_verbose: "product found" }

Query

Key	Value	Description
fields	attributes_en	

Knowledge Panels API | GET https://fr-en.openfoodfacts.org/api/v2/product/00434034/swiss-chocolate-extra-fine-milk-marks-spencer

Description

The Knowledge Panel API is currently a work in progress, aimed at simplifying information display for apps. It will be documented here once it's ready.

The URL provided is just for courtesy, do not assume the response will be stable.

Query

Key	Value	Description
fields	knowledge_panels	

7. Metadata 15

This section describes metadata that are generated by Open Food Facts. Those are often static data that you might need down the road, like a list of all allergens, ingredients, countries, languages, nova groups, etc...

Taxonomies

A taxonomy is a regulated syntax definition for a property; for example, allergens. The definition includes all possible entries and translations into other languages (synonyms). Taxonomies are global and multilingual and do not vary by country.

The taxonomy file is static, it is created when a new taxonomy is built, it is stable and validated by the OFF team.

Taxonomies are not considered API calls, since they are static files.

`https://world.openfoodfacts.org/data/taxonomies/allergens.json`

The product's category parents are indicated in the first line of the taxonomy:

```
en:chocolate-advent-calendars: {
parents: [
"en:advent-calendars",
"en:christmas-chocolates"
],
```

Facets

A facet refers to all the values that contributors add to a property. A facet includes the values defined in the taxonomy and the new values added by the contributors. Facet change constantly and their values are not validated. Facets vary by country.

Facet queries can be made to retrieve a list of the values that belong to a specific facet (for example, labels) and its product count.

A facet query has the following structure:

`https://world.openfoodfacts.org/allergens.json`

You can replace `world` with any of the country codes described in the **Countries** taxonomy.

The values of the facet that are not included in the taxonomy are marked with an asterisk (*).

See an example here: https://us.openfoodfacts.org/labels

Categories

A category is a "tag" used to classify foods in different groups. For example, cheeses. Categories can be freely entered by users. Food category is one of the facets of Open Food Facts. Other examples are allergens or additives.

Note that there is also a taxonomy of categories used to define as many as possible of the “tags” entered by users as known entries in the taxonomy.

The following query retrieves a list of all categories available:

```
https://world.openfoodfacts.org/categories.json
```

You can retrieve a list of products that belong to a specific category. For example, “cheeses”:

```
https://world.openfoodfacts.org/category/cheeses.json
```

Note that the query has an additional parameter “category”.

Important! The categories hierarchy is not a tree but a lattice: each node can have several children, but also several parents.

List Allergens | GET https://world.openfoodfacts.org/allergens.json

Description

List Additives | GET https://world.openfoodfacts.org/additives.json

Description

Query **allergens** facet.

List Additive classes | GET https://world.openfoodfacts.org/data/taxonomies/additives_classes.json

See examples below for taxonomy and other queries.

Description

Query **additives** facet.

Select Example Request/Response

See examples below for taxonomy and other queries.

Query **additives classes** taxonomy.

The response includes the code and name of the additive, a link to a Wikipedia page with more information about the additive, and the number of products containing this additive in the Open Food Facts database.

Select Example Request/Response

Headers

Key	Value	Description
Content-Type	application/json	

Select Example Request/Response

List Brands | GET https://world.openfoodfacts.org/brands.json

Description

Query **brands** facet.

See examples below for taxonomy and other queries.

Select Example Request/Response

List Countries | GET https://world.openfoodfacts.org/countries.json

Description

Query **countries** facet.

See examples below for taxonomy and other queries.

The only country code accepted for queries to the API is ‘country_code_2’. The other formats are only provided for your convenience. Those are the country codes for Top Level Domains.

Select Example Request/Response

List Ingredients | GET https://world.openfoodfacts.org/ingredients.json

Description

Query **ingredients** facet.

See examples below for taxonomy and other queries.

Select Example Request/Response

List Ingredients Analysis | GET https://world.openfoodfacts.org/data/taxonomies/ingredients_analysis.json

Description

Query **ingredients** taxonomy.

This request is used to get information about absence or unawareness of the presence of:

- **palm oil**: `Palm oil free`, `Palm oil`, `Palm oil content unknown`, `may-contain-palm-oil`
- **vegetarian ingredients**: `vegetarian`, `non-vegetarian`, `vegetarian-status-unknown`, `maybe-vegetarian`.
- **vegan ingredients**: `vegan`, `non-vegan`, `vegan-status-unknown`, `maybe-vegan`.

Important! Parsing might not be perfect and the ingredient detection might have issues in some languages. For more information on how the translation works, see: https://github.com/openfoodfacts/openfoodfacts-server/blob/master/taxonomies/ingredients.txt

Select Example Request/Response

List Languages | GET https://world.openfoodfacts.org/languages.json

Description

Query **languages** facet.

See examples below for taxonomy and other queries.

Select Example Request/Response

List Nova Groups | GET https://world.openfoodfacts.org/data/taxonomies/nova_groups.json

Description

Query **nova groups** taxonomy.

Select Example Request/Response

List Nutrients | GET https://world.openfoodfacts.org/cgi/nutrients.pl

Description

Open Food Facts uses optical character recognition (OCR) to retrieve nutritional data and other information from the product labels.

Process

1. Capture the barcode of the product where you want to perform the OCR.
2. The Product Opener server software opens the image (`process_image=1`)
3. Product Opener returns a JSON response. Processing is done using Tesseract or Google Cloud Vision (recommended). The result is often cripped with errors with Tesseract, less with Google Cloud Vision.

Notes: * The OCR may contain errors. Encourage your users to correct the output using the ingredients WRITE API. * You can also use your own OCR, especially if to plan to send a high number of queries.

OCR with Google Cloud Vision

We recommend Google's Vision API to detect and extract text from the images.

For more information about this product, see: https://cloud.google.com/vision/docs/ocr?hl=en

Set `ocr_engine=google_cloud_vision` to use it.

Query

Key	Value	Description
code	04963406	
id	ingredients_fr	
process_image	1	
ocr_engine	google_cloud_vision	

Select Example Request/Response

List Nutrient Levels | GET https://world.openfoodfacts.org/data/taxonomies/nutrient_levels.json

Description

Query **nutrient levels** taxonomy.

The nutrient levels indicate the quantity of fat, saturated fat, sugar and salt in a product.

The quantity levels are the following:

- **low**
- **moderate**
- **high**

For more information about the quantity levels, read the annex 3 of the guide on the development of front of pack nutrition labels, issued by the Department of Health of the British Government, the Food Standards Agency, and devolved administrations in Scotland, Northern Ireland and Wales in collaboration with the British Retail Consortium. Annex 3. Determining red, amber and green colour coding (and High, Medium and Low (HML) text if applied): https://www.food.gov.uk/sites/default/files/media/document/fop-guidance_0.pdf

Examples:

- Saturated fat in moderate quantity
- Salt in high quantity

Select Example Request/Response

List States | GET https://world.openfoodfacts.org/states.json

Description

You can use the following query to retrieve the states taxonomy:

```
GET https://world.openfoodfacts.org/data/taxonomies/states.json
```

Use the following query to retrieve the states facet:

```
GET https://world.openfoodfacts.org/states.json
```

You can drill-down to the list of products in a certain state by making the following call:

```
GET https://world.openfoodfacts.org/state/statename.json
```

Example:

To retrieve a list of products with photo, you can make the following request:

```
GET https://world.openfoodfacts.org/state/photos-uploaded.json
```

Select Example Request/Response

List stores | GET https://world.openfoodfacts.org/stores.json

Description

Query the stores taxonomy

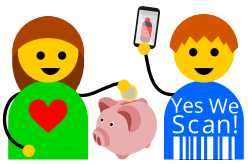
List origins | GET https://world.openfoodfacts.org/origins.json

List packaging | GET https://world.openfoodfacts.org/packagings.json

8. Developer Journeys 6

Meet Dave.

Dave is an active Open Food Facts **contributor** and a **developer** who wants to build **HealthyFoodChoices**, an Android app aimed at conscious consumers that buy healthy products.



HealthyFoodChoices will query Open Food Facts API and provide information on healthy foods available in the place users are living in. Users can narrow down the results by applying different filters and save their search criteria so that the app shows them the products that match their preferences next time they use it.

To identify the potential users' needs, Dave has met with some conscious consumers.

- **Anna** is a 25-year old New Yorker who **doesn't drink soda**, but **her nephew does**. She wants to **compare the nutrition facts of two cola brands**, and **its variants** (`diet`, `zero`, and so on) to decide which one to buy.
- **Stefano** is a 36-year old Italian who follows a **plant-based diet** and wants to **avoid the intake of palm oil**. He's looking for a **breakfast cereal brand** that does not use **palm oil** nor **additives** and has a **great nutriscore (A)**.

Dev Journey 1: Comparing sodas for Anna | GET

Description

Dave wants his app to make an API call to provide Anna the information she needs to make a conscious choice when buying sodas.

Authentication and Header

To make the API query that returns the products that might be interesting for Anna, Dave doesn't need to authenticate (`READ` request).

However, he has to add a `User-Agent` HTTP Header with the name of his app, the version, system and a url (if any), so that he doesn't get blocked by mistake.

In this case, that would be: `User-Agent: HealthyFoodChoices - Android - Version 1.0`

Subdomain

Since Anna lives in NY, Dave wants to define the subdomain for the query as `us`. The subdomain automatically defines the country code (`cc`) and language of the interface (`lc`).

The country code determines that only the products sold in the US are displayed. The language of the interface for the country code `us` is English.

In this case:

https://us.openfoodfacts.org

API Version

The current version number of the Open Food Facts API is v0.

https://us.openfoodfacts.org/api/v0

Product Barcode

After the version number, the word "product", followed by its barcode must be added:

https://us.openfoodfacts.org/api/v0/product/

The app will provide Anna with information about additives, sugars and nutriscore of different types of colas, to help her make her purchase decision.

Anna selects the products she wants to compare in the application (Coca-Cola, Pepsi, Coca-Cola diet, Coca-Cola zero and Pepsi diet). The app retrieves the corresponding barcodes and makes the following calls:

- Pepsico Pepsi Cola Soda: https://us.openfoodfacts.org/api/v0/product/01223004
- Coca-Cola Classic Coke Soft Drink https://us.openfoodfacts.org/api/v0/product/04963406
- Diet Pepsi https://us.openfoodfacts.org/api/v0/product/069000019832
- Coca-Cola Zero https://us.openfoodfacts.org/api/v0/product/5000112519945

Dev Journey 2: Finding healthy breakfast cereals for Stefano | GET

Description

Dave wants his app to make an API call to provide Stefano healthy plant-based breakfast cereals.

Authentication and Header

To make the API query that returns the products that might be interesting for Anna, Dave doesn't need to authenticate. However, he has to add a User-Agent HTTP Header with the name of his app, the version, system and a url (if any), not to be blocked by mistake.

In this case, that would be: `User-Agent: HealthyFoodChoices - Android - Version 1.0`

Subdomain

Since Stefano lives in Italy, Dave wants to define the subdomain for the query as us. The subdomain automatically defines the country code (`cc`) and language of the interface (`lc`).

The country code determines that only the products sold in the Italy are displayed. The language of the interface for the country code `it` is Italian.

In this case:

https://it.openfoodfacts.org

Query Parameters

Dave wants to fine-tune the query to provide Anna with the products that match her buying preferences. To do so, he wants to drill down the results to display only breakfast cereals.

First, he adds the following sequence after the https call: `/cgi/search.pl?` (all search queries need to include this)

Then, he defines some tags and the appropriate values: `action=process&tagtype_0=categories&tag_contains_0=contains&tag_0=breakfast_cereals`

where:

- `action` introduces the action to be performed (process)
- `tagtype_0` adds the first search criterion (categories)
- `tag_contains_0=contains` determines that the results should be included (note that you can exclude products from the search)
- `tag_0` defines the category to be filtered by (breakfast_cereals)

Note: The parameters are concatenated with `&`.

To retrieve breakfast cereals sold in the US, Dave makes the following: https://us.openfoodfacts.org/cgi/search.pl?action=process&tagtype_0=categories&tag_contains_0=contains&tag_0=breakfast_cereals

With this query, the nutrition facts of more than 200 products are displayed.

Then, Dave wants to exclude the products that contain ingredients from palm oil. He adds a new parameter to the query:

- `ingredients_from_palm_oil=without`

This parameter excludes the products that might contain palm oil ingredients from the search.

https://us.openfoodfacts.org/cgi/search.pl?action=process&tagtype_0=categories&tag_contains_0=contains&tag_0=breakfast_cereals&ingredients_from_palm_oil=without

Next, Dave adds another parameter to exclude the products that contain additives:

- `additives=without`

The query is as follows:

https://us.openfoodfacts.org/cgi/search.pl?action=process&tagtype_0=categories&tag_contains_0=contains&tag_0=breakfast_cereals&ingredients_from_palm_oil=without&additives=without

Finally, Dave adds another parameter to include only products with a nutriscore A. The nutriscore is a nutrition grade determined by the amount of healthy and unhealthy nutrients.

- `tagtype_1=nutrition_grade`
- `tag_contains_1=contains`
- `tag_1=A`

The complete query looks like this:

https://us.openfoodfacts.org/cgi/search.pl?action=process&tagtype_0=categories&tag_contains_0=contains&tag_0=breakfast_cereals&tagtype_1=nutrition_grades&tag_contains_1=contains&tag_1=A&additives=without&ingredients_from_palm_oil=without

Add the json=true parameter to avoid scraping.

https://us.openfoodfacts.org/cgi/search.pl?action=process&tagtype_0=categories&tag_contains_0=contains&tag_0=breakfast_cereals&tagtype_1=nutrition_grades&tag_contains_1=contains&tag_1=A&additives=without&ingredients_from_palm_oil=without&json=true

Anna can see now at a glance which products match her search criteria. In this case, around 20 brands of breakfast cereals.

Dev Journey 3: Adding missing products | GET

Description

Dave regularly adds new products to the database and completes missing information via API calls. He has described the process below to show other developers how easy it is to contribute.

Authentication and Header

If you have an app that makes POST calls and you don't want your users to authenticate in Open Food Facts, you can create a global account. Dave has created a global account for the app he is developing with the following credentials:

- `user_id`: myappname
- `password`: 123456

Subdomain

Dave wants to define the subdomain for the query as `us`. The subdomain automatically defines the country code (`cc`) and language of the interface (`lc`).

The country code determines that only the products sold in the US are displayed. The language of the interface for the country code US is English.

In this case:

https://us.openfoodfacts.org/cgi/product_jqm2.pl?

Product Barcode

After the version number, the word `code`, followed by its barcode must be added:

https://us.openfoodfacts.org/cgi/product_jqm2.pl?code=0074570036004

Credentials

Dave adds his user credentials to the call as follows:

https://us.openfoodfacts.org/cgi/product_jqm2.pl?code=0074570036004&user_id=myappname&password=******

Parameters

You can define one or more parameters to add, for example, the brand and the Kosher label:

- `brands`: Häagen-Dazs
- `labels`: kosher

The call looks like this:

```
POST https://us.openfoodfacts.org/cgi/product_jqm2.pl?code=0074570036004&user_id=test&password=test&brands=Häagen-Dazs&labels=kosher
```

Adding a Comment to your WRITE request.

Use the `comment` parameter to add the id of the user editing the product. The id should not contain any personal data.

Important! The user id is not the identifier of an Open Food facts user, but the id generated by your system.

It should be structured as: user-agent + user-id.

Example

```
comment=Edit by a Healthy Choices 1.2 iOS user - SxGFRZkFwdytsK2NYaDg4MzRVenNvUEI4LzU2a2JWK05LZkFRSWc9PQ
```

Adding Additional Information to Existing Fields

To add additional information to existing parameters, add the prefix `add_` to the parameter name.

Important! If you don't use the `add_` prefix, the existing values will be deleted.

Example

```
POST https://us.openfoodfacts.org/cgi/product_jqm2.pl?code=0074570036004&user_id=test&password=test&add_categories=Desserts
```

To see the complete list of parameters, see the **Parameters** section.

Dev Journey 4: Get the Nutri-Score | GET

Description

- If you can't get the information on a specific product, you can get your user to send photos and data.
- That will then be processed by Open Food Facts to get the computed result you want to show them.
- You can implement the complete flow so that they get immediately the result with some effort on their side.
- That will ensure user satisfaction

https://docs.google.com/document/d/1_Y3tdqB8w3VkL6tXqjzPVmkmFiXsBfy0UfS6Mq3MkLs/edit

Select Example Request/Response

Dev Journey 5 : Get the Eco-Score | GET

Description

- If you can't get the information on a specific product, you can get your user to send photos and data.
- That will then be processed by Open Food Facts to get the computed result you want to show them.

- You can implement the complete flow so that they get immediately the result with some effort on their side.
- That will ensure user satisfaction

https://docs.google.com/document/d/1_5AeofpXbaKY9Rd3eeWmHIrhJE8GiPQ-Mfx1SCvpzME/edit?usp=sharing

Dev Journey 6: Get ingredient related analysis on new or existing products (Nova, allergens, additives...) | GET

Description

- If you can't get the information on a specific product, you can get your user to send photos and data.
- That will then be processed by Open Food Facts to get the computed result you want to show them.
- You can implement the complete flow so that they get immediately the result with some effort on their side.
- That will ensure user satisfaction

https://docs.google.com/document/d/1avnxJr8_m6OjRBt0vgwBzlzaZB7Q6z14t0taMKIrkp0/edit

9. FAQ 1

This section provides answers to frequently asked questions.

Some parameters allow the use of `1` and `true` (e.g. `json`). Which one should I use?

Although both `1` and `true` are supported, we recommend using `true`. `1` still works, but will be deprecated soon.

Is multi-filtering supported?

No. You cannot search for different products in the same request:

searchUrl....+ "&tagtype_0=nutrition_grades&tag_contains_0=contains&tag_0=CProducts" + "&tagtype_1=nutrition_grades&tag_contains_1=contains&tag_1=Products"

The boolean logic behind requests is AND.

Can I search for a specific writing of an ingredient? (for example: en:strawberry and not en:strawberries?)

Yes. You can do it with the search box: https://world.openfoodfacts.org/cgi/search.pl?search_terms=strawberries&search_simple=1&action=process. Note that this also searches the product name and brand, not only the ingredients.

You can use the MongoDB export or CSV export. The API is ONLY for scan apps: 1 real user action = 1 API call.

Is there a difference between `nutrition-score-fr_100g` and `nutrition-score-uk_100g`?

Yes, these parameters refer to different concepts.

`nutrition-score-fr_100g` : experimental nutrition score derived from the UK FSA score and adapted for the French market (formula defined by the team of Professor Hercberg).

Nutri-Score definition: a synoptic 5-colour system put forward by public-health researchers (a graphic improvement of the "5-C" system put forward in November 2013 by Serge Hercberg). It places products into five categories set up on the basis of a score that describes the nutritional quality of the product based on its content in major nutrients and certain other items; the five colours appear systematically on the packaging, with a "magnifying glass" on the one that relates to the product.

The nutritional score is calculated using the data from the nutritional declaration for 100 g of the product as sold. With liquid foods such as soups, oils or milk, the value used is the one given as a unit on the nutrition label, and not a unit that is not written on the packaging (to ensure transparency for the consumer). If two values are mentioned (per 100 g and per 100 ml), the one per 100 g is to be taken into account. Source: https://www.santepubliquefrance.fr/media/files/02-determinants-de-sante/nutrition-et-activite-physique/nutri-score/qr-scientifique-technique-en

For more information, see: https://solidarites-sante.gouv.fr/IMG/pdf/rapport_etiquetage_nutritionnel_version_anglaise.pdf

`nutrition-score-uk_100g` : nutrition score defined by the UK Food Standards Administration (FSA)

For more information, see: * https://www.gov.uk/government/publications/the-nutrient-profiling-model * https://*.openfoodfacts.org/nutriscore * https://*.openfoodfacts.org/nova * https://*.openfoodfacts.org/nutrient-levels

---

Other Sources of Information

Open Food Facts contains only information about packaged food. For average values of produce (for example, tomatoes or bananas) and other food products, you can use one of the official national nutrition databases instead.

Note: The list below contains some of the most important national food databases. If you think some other database should be included in the list, please contact us at: https://world.openfoodfacts.org/contact

List of National Food Databases

- **Australia** - FSANZ - NUTTAB 2006: https://www.foodstandards.gov.au/media/documents/FSANZ%20Conf%20PostersNUTTAB.pdf
- **Belgium** - NUBEL - Belgian Food Composition Data: https://www.internubel.be
- **Canada** - FCEN: https://aliments-nutrition.canada.ca/cnf-fce/index-fra.jsp
- **Czech Republic** - Food Composition Database at National Institute of Public Health: http://www.chpr.szu.cz/dbdata/foodcomp/nut2001.asp
- **Denmark** - Danish Food Composition Databank: https://frida.fooddata.dk/?lang=en
- **Estonia** - Estonian Food Composition Database: https://tka.nutridata.ee/en/
- **Finland** - Finnish Food Composition Database - FINELI: https://fineli.fi/fineli/en/index
- **France** - CIQUAL: https://www.anses.fr/en/search/site/Table%20ciqual
- **Germany** - Souci-Fachmann-Kraut Online Database: https://www.sfk.online/#/home
- **Italy** - Banca Dati di Composizione degli Alimenti CREA: https://www.crea.gov.it/web/alimenti-e-nutrizione/banche-dati
- **Netherlands** - Dutch Food Composition Database: https://www.rivm.nl/en/dutch-food-composition-database
- **Norway** - The Norwegian Food Composition Table 2006: https://www.matvaretabellen.no/?language=en
- **Poland** - Food Composition Tables: http://www.izz.waw.pl/en/?lang=en
- **Spain** - Spanish Food Composition Database - BEDCA: https://www.bedca.net/bdpub/index.php
- **Switzerland** - Swiss Food Composition Database: https://www.naehrwertdaten.ch/de/
- **UK** - Composition of foods integrated dataset (CoFID): https://www.gov.uk/government/publications/composition-of-foods-integrated-dataset-cofid
- **USA** - USDA: https://ndb.nal.usda.gov/

String Normalization

The normalization process is different depending on the language:

- `no_language` is used for strings that are not in a specific language (e.g. user names)
- `default` is used for languages that do not have specified values
- German: Umlauts are not converted (e.g. ä -> ae)
- All languages: the parameters are converted to lowercase and unaccented. The following special characters are converted:
  - [à|á|â|ã|ä|å] -> a
  - [ç] -> c
  - [è|é|ê|ë] -> e
  - [ì|í|î|ï] -> i
  - [ñ] -> n
  - [ò|ó|ô|õ|ö] -> o
  - [ù|ú|û|ü] -> u
  - [ý|ÿ] -> y
  - [œ|Œ] -> oe

- [æ|Æ] -> ae

- Punctuation signs are changed to a dash -

Example of the normalization process for the product: `coffee`, brand: `Nescafé`

Non-taxonomized fields:

- `brands`: Nescafé (as typed, no normalization)
- `brands_tags`: nescafe (normalized = lower-case, unaccented, punctuation signs are changed to a dash -)

Taxonomized fields:

- `categories`: Café
- `categories_tags`: en:coffees

In this case, the tags are an id in the relevant taxonomy.

WRITE API: * Always use the raw unprocessed value (Nescafé). Do not try to provide the tag directly (taxonomized or not).

READ API: * If the field is taxonomized, use the taxonomy file to translate the `_tag` value into the user's native language (see the **Taxonomies** section).

- If the field is not taxonomized, use the raw unprocessed value.

Which products are considered beverages in the NutriScore?

The following products are not considered beverages:

```
en:plant-milks
en:milks
en:dairy-drinks
en:meal-replacement
en:dairy-drinks-substitutes
en:chocolate-powders
en:soups
en:coffees
en:tea-bags
en:herbal-teas
```

The following products are considered beverages:

```
en:tea-based-beverages
en:iced-teas
en:herbal-tea-beverages
en:coffee-beverages
en:coffee-drinks
```

For more information, see: https://world.openfoodfacts.org/nutriscore

---

Which products are not taken into account for the NutriScore?

```
en:alcoholic-beverages
en:aromatic-herbs
en:baby-foods
en:baby-milks
en:chewing-gum
en:coffees
en:food-additives
en:herbal-teas
en:honeys
en:meal-replacements
en:salts
en:spices
en:sugar-substitutes
en:vinegars
en:pet-food
en:non-food-products
```

The information below has been taken from the Nutri-Score FAQ document, available online at: https://www.santepubliquefrance.fr/content/download/150263/file/QR_scientifique_technique_EN_011119.pdf

Food products that are not covered by the mandatory nutritional declaration are listed in Appendix V of regulation no. $^{1169}/_{2011}$. They are:

1. **Unprocessed products** that comprise a single ingredient or category of ingredients (such as fresh fruits or vegetables, cut raw meat, honey, etc.)
2. **Processed products** where the only processing they have been subjected to is maturing and that comprise a single ingredient or category of ingredients Note: here the products in question are mainly meat products
3. **Waters** intended for human consumption, including those where the only added ingredients are carbon dioxide and/or flavourings
4. **Herbs**, **spices** or mixtures thereof
5. **Salt** and salt substitutes
6. Table top **sweeteners**
7. Products covered by **Directive 1999/4/EC of the European Parliament** and of the Council of 22 February 1999 relating to coffee extracts and chicory extracts, whole or milled coffee beans, and whole or milled decaffeinated coffee beans
8. **Herbal and fruit infusions**, tea, decaffeinated tea, instant or soluble tea or tea extract, decaffeinated instant or soluble tea or tea extract, which do not contain other added ingredients than flavourings which do not modify the nutritional value of the tea.
9. **Fermented vinegars** and substitutes for vinegar, including those where the only added ingredients are flavourings.
10. **Flavourings**
11. **Food additives**
12. **Processing aids**
13. **Food enzymes**
14. **Gelatine**
15. **Jam setting compounds**
16. **Yeasts**
17. **Chewing gums**
18. Food in **packaging** or **containers** the largest surface of which has an area of **less than 25 cm²**
19. Food, including handcrafted food, **directly supplied by the manufacturer of small quantities** of products to the final consumer or to local retail establishments directly supplying the final consumer.

I've found a bug in the API, but I'm not sure if the issue has been already reported. Where can I find a list of existing issues? How can I create a new bug fix request?

Before creating a new bug fix request, make sure the issue has not been reported yet. The following link displays a full list of issues for the backend (in different states):

https://github.com/openfoodfacts/openfoodfacts-server/issues?utf8=%E2%9C%93&q=is%3Aissue+is%3Aopen+label%3Aapi+

To report a bug, inform us in the API channel o create a bug fix request on GitHub: https://github.com/openfoodfacts/openfoodfacts-server/blob/master/CONTRIBUTING.md

---

How can I set the main language of a product?

The `lang` parameter allows you to set the main language of the product. If not explicitly defined, the main language will be the first language added to the product.

In the case of a multilingual product, you can specify the main language of the product, and you can then specify values and images for different languages by suffixing the language code to the other fields.

Examples:

- `lang=fr`
- `ingredients_text_with_allergens_fr`

About Salt and Sodium

`salt` is automatically converted to `sodium` and vice-versa. Both values are stored in the database. Note that, if you want to delete `sodium`, `nutriment_salt` has to be deleted as well.

dummy | GET

Robotoff API 10

About

The Robotoff project is intended to complete missing information of products by prompting users to confirm predictions inferred by Artificial Intelligence algorithms. These algorithms are calculated based on “insights”, which are facts about a product that have been extracted or deduced from the product pictures, ingredients, categories, labels, etc...

The project URL is: https://robotoff.openfoodfacts.org/api/v1/{endpoint}.

Robotoff can interact with all Open Food Facts products and environments. The `server_domain` field must be used to specify the product/environment (api.openfoodfacts.org for OFF-prod).

Configuration

To configure this feature in your app follow the steps below:

1. Fetch a JSON file when opening a product. Example: https://robotoff.openfoodfacts.org/api/v1/questions/3274570800026?lang=en&count=3

   {"questions": [{"barcode": "3274570800026", "type": "add-binary", "value": "Scallop", "question": "Does the product belong to this category?", "insight_id": "5cac03bc-a5a7-4ec2-a548-17fd9319fee7", "insight_type": "category", "source_image_url": "https://static.openfoodfacts.org/images/products/327/457/080/0026/front_en.4.400.jpg"}], "status": "found"}

2. Display the question and possible answers in the UI.
3. Send back the proper ping to the Open Food Facts server if the user answers.

   https://github.com/openfoodfacts/robotoff/blob/master/doc/api.md

   http://robotoff.openfoodfacts.org/api/v1/insights/annotate?insight_id=(insight_id)&annotation=(1,0,-1)&update=1

Get a random insight | GET https://robotoff.openfoodfacts.org/api/{{ROBOTOFF_API_VERSION}/questions/random

Description

Fetch a random insight.

Parameters

- `type` (string, optional): The type of insight. If not provided, an insight from any type will be returned.
- `country` (string, optional): Only return predictions with products from a specific country (ex: en:france)
- `value_tag` (string, optional): Filter by value tag, i.e the value that is going to be sent to Open Food Facts.
- `server_domain` (string, optional): Server domain. Default to ‘api.openfoodfacts.org’

Query

Key	Value	Description
lang	fr	
insight_types	category	str, optional - comma-separated list, the type of insight. If not provided, an insight from any type will be returned.
server_domain	api.openfoodfacts.org	str, optional - server domain. Default to ‘api.openfoodfacts.org’
count	10	str, optional - number of results to return (default: 1)
value_tag	en:bcaa	str, optional - filter by value tag, i.e the value that is going to be sent to Openfoodfacts
country	en:france	str, optional - Only return predictions with products from a specific country (ex: en:france)
brands	ironmaxx	(string, optional): filter by brands, comma-separated list of brand tags.

Select Example Request/Response

Get insights (filtering system) | GET https://robotoff.openfoodfacts.org/api/{{ROBOTOFF_API_VERSION}/insights/

Description

Return all insights associated with a specific product.

Parameters

- `barcode`: Product barcode
- `server_domain` (string, optional) - server domain. Default to ‘api.openfoodfacts.org’

Query

Key	Value	Description
barcode	0021000123803	Optional. Allows to get all insights for a product
lang	fr	The language the response is served in, useful to get translated questions to ask users

Key	Value	Description
server_domain	api.openfoodfacts.org	Domain it’s queried from
count	50	number of insight you’d like
insight_types	packaging	packaging, category, label, brand,product_weight
page	1	In some cases, you might want the nth page of insights

Get a specific insight | GET https://robotoff.openfoodfacts.org/api/{{ROBOTOFF_API_VERSION}/insights/detail/

Description

Parameters

- `insight_id`: ID of the insight

Query

Key	Value	Description
id	23541d80-02fc-4cd6-88eb-d93aa17e3386	ID of the insight

Submit an annotation | POST https://robotoff.openfoodfacts.org/api/{{ROBOTOFF_API_VERSION}/insights/annotate

Description

Submit an annotation, given the `insight_id`. The request type must be application/x-www-form-urlencoded.

Parameters

- `insight_id` (string, required): ID of the insight
- `annotation` (integer, required): Annotation of the prediction:
  - 1 to accept the prediction
  - 0 to refuse it
  - -1 for “unknown”.
- `update` (integer, optional): Send the update to Open Food Facts if update=1. Otherwise, the update won’t be sent. This parameter is useful if the update is performed client-side.

Query

Key	Value	Description
insight_id	23541d80-02fc-4cd6-88eb-d93aa17e3386	(str, required) - ID of the insight
annotation	0	(int, required) - Annotation of the prediction: 1 to accept the prediction, 0 to refuse it, and -1 for “unknown”.
update	false	(int, optional) - Send the update to Openfoodfacts if update=1, don’t send the update otherwise. This parameter is useful if the update is performed client-side.

Get questions | GET https://robotoff.openfoodfacts.org/api/{{ROBOTOFF_API_VERSION}/questions/

Description

You can get questions for a given product or get random questions.

Parameters to be used to get questions for a given product

- `barcode`: Product barcode
- `lang` (string, optional): the language of the question/value. **Default**: en.
- `count` (integer, optional): Number of questions to return. **Default**: 1.
- `server_domain` (string, optional): server domain. **Default**: ‘api.openfoodfacts.org’

Parameters to be used to get random questions

- `lang` (string, optional): the language of the question/value. **Default**: en.
- `count` (integer, optional): Number of questions to return. **Default**: 1.
- `insight_types` (list, optional): comma-separated list, filter by insight types.
- `country` (string, optional): filter by country tag.
- `brands` (string, optional): filter by brands, comma-separated list of brand tags.
- `value_tag` (string, optional): filter by value tag, i.e the value that is going to be sent to Openfoodfacts.
- `server_domain` (string, optional): server domain. **Default**: api.openfoodfacts.org.

Query

Key	Value	Description
lang	fr	(str, optional) - the language of the question/value. ‘en’ by default.
count	10	(int, optional) - Number of questions to return. Default to 1.
server_domain	api.openfoodfacts.org	(str, optional) - server domain. Default to ‘api.openfoodfacts.org’
barcode	0021000123803	Product barcode

Get statistics for a user | GET https://robotoff.openfoodfacts.org/api/{{ROBOTOFF_API_VERSION}/users/statistics/

Query

Key	Value	Description
username		Open Food Facts username

Get API status | GET https://robotoff.openfoodfacts.org/api/{{ROBOTOFF_API_VERSION}/status

Import image predictions | GET https://robotoff.openfoodfacts.org/api/{{ROBOTOFF_API_VERSION}/images/predictions/import

Image Crop (Robotoff side) | GET https://robotoff.openfoodfacts.org/api/{{ROBOTOFF_API_VERSION}/images/crop

Query

Key	Value	Description
image_url	https://static.openfoodfacts.org/images/products/317/718/000/0810/1.jpg	
y_min	0.758063614	
x_min	0.888398051	
y_max	0.993165255	
x_max	0.994514585	

Get insights for popular products | GET https://robotoff.openfoodfacts.org/api/v1/questions/popular

Description

This API was a Christmas present from Raphael to Pierre

Query

Key	Value	Description
count	5	

Generated at 2022-03-29 08:39:22 by docgen