User guide

BigBuy API

In this guide, you will find the basic aspects to learn in order to use BigBuy's API

Contents

Introduction

Thanks to this initial guide, you will be able to find out how BigBuy's API is structured to make it easier for you to use.

Before continuing, we recommend you familiarise yourself with our API, in addition to its technical documentation at the following link.

Reference material for the API: https://api.bigbuy.eu/rest/doc

Authentication

All requests to BigBuy's API must contain the following header:

Authorization: Bearer *API_KEY*

API_KEY is provided upon request using the contact form.

The base URL of the API is https://api.bigbuy.eu and https://api.sandbox.bigbuy.eu for the test API.

Obtain BigBuy's catalogue

BigBuy's catalogue contains all of the information and structure of all of its products.

As well as the products, we have the following relevant entities:

- **taxonomies**: the products are classified by taxonomies (categories)
- **tags**: tags of the product
- **manufacturers**: information about product manufacturers
- **attributes/attributesGroup**: attributes of products
- **images:** product images
- **stock:** stock of all products and variations
- **variations:** product variations

Product taxonomies

All of BigBuy's products are classified in a taxonomy. To obtain the full tree of taxonomies, we will use the following endpoint:

GET /rest/catalog/taxonomies.json

Response

```
[
   {
       "id": 2,
       "name": "Acampada y senderismo",
       "url": "deportes-y-aire-libre-acampada-y-senderismo",
       "parentTaxonomy": 19756,
       "dateAdd": "2021-10-20 13:53:25",
       "dateUpd": "2023-03-01 16:18:08",
       "urlImages": "https://cdnbigbuy.com/images/8435310192985_R02.jpg",
       "isoCode": "es"
   },
   ...
]
```

In order to obtain only first level taxonomies (those situated at the root of the tree of taxonomies), we will provide the **firstLevel** parameter:

GET /rest/catalog/taxonomies.json?firstLevel

With these first level taxonomies we will be able to use the rest of the calls from the catalogue by only filtering the products classified within one of these first level taxonomies.

Obtain products

Once the first level taxonomies have been obtained, we will be able to obtain the BigBuy products corresponding to each of the taxonomies until the catalogue is complete.

GET /rest/catalog/products.json?parentTaxonomy=ID_TAXONOMY

Response

```
[
   {
       "manufacturer": 2976,
       "id": 216701,
       "sku": "S4600179",
       ean13" "7613036867405",
       "weight": 0,355,
       "height": 15,
       "width": 15,
       "depth": 15,
       "dateUpd": "2023-02-27 18:40:19",
       "category": 3157,
       "dateUpdDescription": "2023-02-27 18:40:19",
       "dateUpdImages": "2021-10-28 17:44:05",
       "dateUpdStock": "2023-02-27 18:40:19",
       "wholesalePrice": 8.98,
       "retailPrice": 9.98,
       "dateAdd": "2023-02-27 18:40:19",
       "video": "0",
       "active": 1,
       "attributes": false,
       "categories": true,
       "images": true,
       "taxRate": 10,
       "taxId": 2,
       "inShopsPrice": 13.31,
       "condition": "NEW",
       "logisticClass": "A",
       "tags": true,
       "dateUpdProperties": null,
       "dateUpdCategories": null,
       "priceLargeQuantities": null,
       "intrastat": "84733080"
   },
   …
]
```

Product Information

To obtain the text information, such as the description, name, etc. the following endpoint will be used. We should specify both the first level taxonomy and the desired language.

GET /rest/catalog/productsinformation.json?isoCode=es&parentTaxonomy=ID_TAX

Response

```
[
   {
       "id": 63272,
       "sku": "S0412092",
       "name": "Coffee Capsules Nescafé Dolce Gusto 98423 Lungo (16 uds)",
       "description": "From now on you can easily and quickly prepare delicious coffee with
the most natural, pure and authentic flavor, thanks to the new <b>Coffee Capsules Nescafé
Dolce Gusto 98423 Lungo (16 uds)</b>!<br><ul><li>Type:
<ul><li>Capsules</li><li>Lungo</li></ul></li><li>Contains: 16 uds</li><li>Quantity: 0,2
kg</li><li>Carbohydrates (g): 0,3</li><li>Proteins (g): 0,2</li><li>Energy (kcal):
1</li></ul>",
       "url": "coffee-capsules-nescafe-dolce-gusto-98423-lungo-16-uds_63272",
       "isoCode": "en",
       "dateUpdDescription": "2022-06-29 14:18:13"
   },
   ...
]
```

Product variations

Now, for the products that have attributes (sizes, colours....), we have to obtain their variations. A variation is just a combination of one or more attributes.

Examples:

Perfume S0554780 has the following variations (only 1 group of attributes):

- S0548923 - Group of attributes: Capacity. Attribute: 100ml
- S0548924 - Group of attributes: Capacity. Attribute: 30ml

T-shirt S1405630 has the following variations (2 groups of attributes):

- S1405665
  - Group of attributes: Colour. Attribute: Green
  - Group of attributes: Size. Attribute: XXL
- S1405659
  - Group of attributes: Colour. Attribute: Red
  - Group of attributes: Size. Attribute: XL
- ...

Each possible variation of a product has its own SKU, EAN and specific price.

For example, ham may vary in price according to the attributes selected, because it has the kg (Kilogrammes) attribute and, according to the Kg selected, we have one price or another.

Request to obtain all of the variations of the catalogue classified within a first level taxonomy:

GET /rest/catalog/productsvariations.json?parentTaxonomy=ID_TAXONOMY

You will see each of them has its own identifier (id) and a product identifier

(product) associated with it.

Response

```
[
    {
        "id": 1135492,
        "sku": "S1455910",
        "ean13": "4899888483472",
        "extraWeight": 2.5,
        "product": 453293,
        "wholesalePrice": 17.35,
        "retailPrice": 21.16,
        "inShopsPrice": 26.45,
        "width": 16,
        "height": 16,
        "depth": 22,
        "priceLargeQuantities": [
            {
                "id": 6257354,
                "quantity": 6,
                "price": 16.61
            },
            {
                "id": 6257355,
                "quantity": 12,
                "price": 16.32
            }
        ],
        "logisticClass": "A",
        "ean13Packaging": null,
        "intrastat": null
    },
    ...
]
```

Product attributes

Request to obtain the relationship between variation and attribute:

GET /rest/catalog/variations.json?parentTaxonomy=ID_TAXONOMY

By obtaining the result, we will check that for each variation id we have the attributes it consists of, identified with its own id.

Response

```
[
    {
        "id": 1116685,
        "attributes": [
            {
                "id": 25237
            }
        ]
    },
    {
        "id": 1116686,
        "attributes": [
            {
                "id": 25238
            }
        ]
    },
    ...
]
```

To complete the attributes of products in our catalogue, we just need to know the name of each one of them in the language we desire.

GET /rest/catalog/attributes.json?isoCode=ISO_CODE&parentTaxonomy=ID_TAX

Response
```
[
   {
       "id": 1898,
       "attributeGroup": 150,
       "name": " 1 - Black 1,5 g",
       "isoCode": EN:
   },
   {
       "id": 1892,
       "attributeGroup": 150,
       "name": " 12 - Black",
       "isoCode": "en"
   }
   ,
   ...
]
```

As you can see, we obtain the id and the name of each attribute, but we also obtain the group of attributes it belongs to. We can see this at:

GET /rest/catalog/attributesgroups.json?isoCode=ISO_CODE&parentTaxonomy=ID_TAX

We will receive a structure with all of the attribute groups and their id and name (name):

Response
```
[
   {
       "id": 2,
       "name": Quantity
       "isoCode": "en"
   },
   {
       "id": 3,
       "name": "Capacity",
       "isoCode": "en"
   },
   ...
]
```

The frequency with which we recommend carrying out the above load/update processes is once a day or every 8 hours.

Product images

In this call, we will be able to obtain all of the information relating to the images of each of the BigBuy products.

To obtain more information about the images assigned to products:

GET /rest/catalog/productsimages.{format}?parentTaxonomy=ID_TAXONOMY

Response:

```
[
   {
       "id": 63272,
       images [
           {
               "id": 1050714,
               "isCover": "TRUE"
               "name": "5011546498423_0_P00",
               "url": "https://cdnbigbuy.com/images/5011546498423_0_P00.jpg",
               "logo": false,
               "whiteBackground": false
           }
       ]
   },
   {
       "id": 63274,
       "images": [
           {
               "id": 1050715,
               "isCover": true,
               "name": "7613031526406_0_P00",
               "url": "https://cdnbigbuy.com/images/7613031526406_0_P00.jpg",
               "logo": false,
               "whiteBackground": false
           }
       ]
   },
   ...
]
```

Stock update

In the stock update, for the same product or variation, we will obtain different types of stock according to their preparation time. This will enable us to choose which stock we are interested in. If we are interested in everything, we will limit ourselves to adding all of the stock for one product to obtain the total stock.

The stock update is divided into two parts:

Product stock

We will receive stock of the product according to its preparation time:

/rest/catalog/productsstockbyhandlingdays.json?parentTaxonomy=ID_TAXONOMY,

We will check that the structure is an array in which each item is identified with its product id (id).

Response

```
[
  {
    "id": 63272,
    "sku": "S0412092",
    "stocks": [
      {
        "quantity": 5,
        "minHandlingDays": 0,  // Stock 0-24h
        "maxHandlingDays": 1,
        "warehouse": 1
      },
      {
        "quantity": 193,
        "minHandlingDays": 1, // Stock 24-48h
        "maxHandlingDays": 2,
        "warehouse": 1
      },
     {
        "quantity": 0,
        "minHandlingDays": 2, // Stock 2-4 días
        "maxHandlingDays": 4,
        "warehouse": 1
      }
    ]
  },
  ...
]
```

Stock variation

We will receive stock from all of the product variations in the catalogue, since the products with variations may have different stock depending on their variation.

/rest/catalog/productsvariationsstockbyhandlingdays.json?parentTaxonomy=ID_TAXONOMY,

We will obtain exactly the same structure as the previous one, except that, in this case, id refers to the variation id.

Response

```
[
  {
     "id": 1116685,
     "sku": "V2200194",
     "stocks": [
        {
           "quantity": 2,
           "minHandlingDays": 0,
           "maxHandlingDays": 1,
           "warehouse": 1
        },
        {
           "quantity": 500,
           "minHandlingDays": 1,
           "maxHandlingDays": 2,
           "warehouse": 1
        }

     ]
  },
  ...
]
```

We recommend carrying out this process every 10-15 minutes to be able to keep stock updated.

How to process orders with the Orders API?

Each client has the option of personalising their shipping, choosing the carriers they want to work with or choosing the desired delivery option.

Order request

```
{
   "order": {
       "internalReference": "12345890194",
       "language": "es",
       "paymentMethod": "bankwire",
       "carriers": [
           {
               "name": "gls"
           },
          {
               "name": "seur"
           }

       ],
       "shippingAddress": {
           "firstName": "John",
           "lastName": "Doe",
           "country": "FR",
           "postcode": "75004",
           "town": "Paris",
           "address": "850 Rue des Juges Consuls",
           "phone": "123456789",
           "email": "email@email.com",
           "comment": ""
       },
       "products": [
           {
               "reference": "H4510100",
               "quantity": 1
           }
       ]
   }
}
```

In the request, we should include the following fields:

- **internalReference** - Your internal identifier for the order
- **language**
- **paymentMethod** - bankwire, paypal, moneybox…
- **carriers** - Contains an array with the carriers available for the order
    - **name** - Carrier name (Typical values: chrono, correos, gls, ups, dhl, tnt, seur, correos international, pallet delivery, dachser, dhl freight, postal service, standard shipment, db schenker)
- **shippingAddress** - Shipping details
    - **firstName -** Recipient's first name
    - **lastName** - Recipient's last name
    - **country**
    - **postcode**
    - **town**
    - **address** - Shipping address
    - **phone** - Recipient's telephone
    - **email** - Recipient's email address
    - **comment** - Additional comments if required
- **products** - Array of products selected
    - **reference** - Product reference
    - **quantity**

With this information, we can now validate an order and register it on BigBuy to be able to process it.

We will send the request described above, first to the "check" endpoint and then to the "create" endpoint if everything has gone well during the check.

- **Check**: validation of the order and return of the total. It does not create the order. If we obtain a positive response, then we will be able to create the order by calling "create".

  POST /rest/order/check/multishipping.{format}

Response

```
{
   "orders": [
       {
          "productReferences": [
             "F1520215"
          ],
          "totalWithoutTaxesAndWithoutShippingCost": 503.8,
          "totalWithoutTaxes": 509.15,
          "total": 616.07,
           "warehouse": 1
       }
   ],
   "errors": []
}
```

- **Create**: once a positive response has been obtained from the previous endpoint (Check), we will be able to register it on BigBuy.

  POST /rest/order/create/multishipping.{format}

Response

```
{
   "orders": [
       {
          "productReferences": [
             "F1520215"
          ],
          "id": "15012345",
           "warehouse": 1,
           "url": "/rest/order/15012345"
       }
   ],
   "errors": []
}
```

If an order request contains products that belong to different warehouses, then as many orders as warehouses will be created.

Orders that are correctly created are displayed under the “orders” field of the response and those with any errors will be displayed under the “errors” field.

Example:

```
{
   "orders": [],
   "errors": [
       {
           "status": 409,
           "code": "ER003",
           "message": "{\"info\":\"Products have no
stock.\",\"data\":{\"skus\":["H1500113"]}}",
       }
   ]
}
```

Two of the most common errors that can arise are that the product is no longer active or the product is not in stock. In both cases, if there is any other product in the catalogue with the same EAN and this is available and with stock, this additional information will be displayed so that the user can try to place the order using the alternative product in the “error_detail” field. It is also important to look carefully at the “condition” of the product.

```
{

   "orders": [],
   "errors": [
       {
           "status": 409,
           "code": "ER003",
           "message": "{\"info\":\"Products have no
stock.\",\"data\":{\"skus\":["H1500113"]}}",
           "productReferences": [
               "H1500113"
           ],
           "warehouse": 1,
           "error_detail": [
               {
                   "requested": "H1500113",
                   "alternatives": [
                       {
                           "reference": "S0110525",
                           "stock": 78,
                           "price": 456.81,
                           "condition": "NEW"
                       }
                   ]
               }
           ]
       }
   ]
}
```

In any case, it is important to first do the **check** call in order to find out in advance whether there will be problems in placing the order. In the event that the order contains products from more than one warehouse and we try to place the order directly with **create**, we could find ourselves in the situation where the order is only partially created.

Example:

```
{

   "orders": [
       {
          "productReferences": [
             "F1520215",
             "V0720216"
          ],
          "id": "15012345",
           "warehouse": 1,
           "url": "/rest/order/15012345"
       }
   ],
   "errors": [
       {
           "status": 409,
           "code": "ER003",
           "message": "{\"info\":\"Products have no
stock.\",\"data\":{\"skus\":["H1500113"]}}",
           "productReferences": [
               "H1500113"
           ],
           "warehouse": 2
       }
   ]
}
```

In this case, the order contains products from 2 warehouses, but it has only been possible to create the first order.

Choosing the carrier for the order

BigBuy will choose the **cheapest carrier from all of those provided in the order creation request**. If we want to exclude a carrier, we will have to leave it off the list of carriers permitted when creating the order.

With BigBuy's shipping API, it is possible to find out all of the possible shipping options (carriers) for an order before creating it. To do so, we provide a set of products as well as destination country and postcode.

POST /rest/shipping/orders.{format}

Request:

```
{
 "order": {
   "delivery": {
     "isoCountry": "SE",
     "postcode": "132 51"
   },
   "products": [
     {
       "reference": "S6483140",
       "quantity": 1
     }
   ]
 }
}
```

Response

```
{
   "shippingOptions": [
       {
           "shippingService": {
               "id": "180",
               "delay": "4-6 days",
               "name": "GLS",
               "transportMethod": "van",
               "serviceName": "GLS"
           },
           "cost": 9.68,
           "weight": 0.05
       },
       {
           "shippingService": {
               "id": "62",
               "delay": "4-5 days",
               "name": "SEUR",
               "transportMethod": "van",
               "serviceName": "SEUR"
           },
           "cost": 12.62,
           "weight": 0.05
       },
       ...
   ]
]
```

The “serviceName” field of each option available will be the one we will be able to provide in the request to create an order within the “carriers” section.

```
{

"order": {
     "internalReference": "12345890194",
     ...
     "carriers": [
         {
             "name": "gls"
         },
         ...
       ],
     ...
   }
}
```

We can obtain the full list of carriers and services for orders in the following endpoint:

GET /rest/shipping/carriers.{format}

In the response, we will find all of the carriers available with their services, in addition to the countries they deliver to, blocked products or taxonomies for that carrier, etc.