Shopitem API

A technical guide to the REST API for managing updates of shopitems

IMPORTANT

The Shopitem API has been updated to a new version.

The new documentation can be found at the following location:

http://cl.beslist.nl/pdf/Beslist%20ShopItemAPI%20V3%20documentation.pdf

This documentation should not be used for any new connections.

The v1 and v2 version of the API wil not be supported in the near future.

30-8-2018

Date: 29-07-2016

Version: 1.7

Team Webshops <teamwebshops@beslist.nl>

Index

Introduction and background

The Beslist.nl platform imports webshop inventory items using feeds. The process of feed creation, downloading and processing all items in a feed is by nature not real-time. Therefore an API has been developed enabling near real-time updates to a number of properties of inventory items. Within the context of this API those webshop inventory items are referred to as 'shopitems' and the API is called the 'Shopitem API'.

The Shopitem API is published with built-in technical API documentation aimed at API client developers. This built-in documentation defines the contracts for the REST endpoints, by specifying input and output and documenting resulting status codes. Included in the documentation is a sandbox environment

This document gives an overview of the conventions used for he Shopitem API and gives a few code examples of how to use the API. It is not exhaustive but should give enough information to start using the API. Just like the API will grow in time and new features and endpoints may be supported in future versions, this documentation will grow and change over time.

1. How to get access to the API and its online docs

Location of online API documentation

The Shopitem API has a single host where both the API and the online documentation can be found. The API is versioned, and the version number is in the urls. The documentation URL is a good starting point because it contains the urls to the API endpoints. All API urls, including documentation use https.

Shopitem API docs: https://shopitem.api.beslist.nl/api/doc

Access to all urls in scope of the Shopitem API is currently granted using a whitelist of allowed client ip adresses. If you think you should have access, but get a 403/Forbidden response, please contact our Content Management department at productfeed@beslist.nl

Documentation for different versions of the API endpoints

API endpoints can co-exist in different versions, meaning that multiple versions of a single endpoint can exist. The most recent version is always the default version. Older versions are labeled as deprecated and will be removed in the future. The online documentation is also available in different versions. The default view that is displayed at the doc location url always shows the documentation for the most recent versions of all API endpoints:

Current docs: https://shopitem.api.beslist.nl/api/doc

There is a special view of the documentation showing only the deprecated endpoints. This shows the endpoint versions that will be taken offline in the future. If your client still used those endpoints this gives a list of calls to be migrated to more recent versions.

Deprecated view: https://shopitem.api.beslist.nl/api/doc/deprecated

Besides being shown in the default and deprecated views, very version of the API has its own view. The url can be composed by postfixing the default url with the API version:

Version v1 docs: https://shopitem.api.beslist.nl/api/doc/v1
Version v2 docs: https://shopitem.api.beslist.nl/api/doc/v2

Endpoint urls

The API has several endpoints that are documented in the online documentation. The urls are all based on a versioned url with an endpoint specific suffix. There are two base urls, one for an authrelated endpoint for getting shop_ids for accessible shops, and one for all the actual API endpoints that deal with shopitems:

Auth:

https://shopitem.api.beslist.nl/auth/v1/shops

Single products:

https://shopitem.api.beslist.nl/product/v1/shops/{shopId}/items/{externalId} (GET)

https://shopitem.api.beslist.nl/product/v2/shops/{shopId}/items/{externalId} (PUT)

Product batches

https://shopitem.api.beslist.nl/product/v2/shops/{shopId}/items (PUT)

https://shopitem.api.beslist.nl/product/v2/shops/{shopId}/batch/{batchUuid
}

Now deprecated:

https://shopitem.api.beslist.nl/product/v1/shops/{shopId}/items/{externalId}/delivery_cost_be

https://shopitem.api.beslist.nl/product/v1/shops/{shopId}/items/{externalId}/delivery_cost_nl

https://shopitem.api.beslist.nl/product/v1/shops/{shopId}/items/{externalId}/delivery_time_be

https://shopitem.api.beslist.nl/product/v1/shops/{shopId}/items/{externalId}/delivery_time_nl

https://shopitem.api.beslist.nl/product/v1/shops/{shopId}/items/{externalId}/price

API access keys

Every call to an endpoint url must be authenticated with an API access key. This key must be sent in a header “apikey”. An access key can be generated for you, but is not stored and cannot be recovered or reverse engineered. If you lose your key you will need to have a new one generated for you. Keep it secured, because it enables making changes to product of your shop!

Getting an API key

To prevent corrupting a shop’s data, we have a sandbox API environment which can be found at: https://test-shopitem.api.beslist.nl/api/doc

Contact productfeed@beslist.nl for access to this environment. They will fill the test-environment with actual data for your webshop and provide you with an API key. Once you have been able to update 70% of the products in your data feed, your request for a live key will be considered.

If you are not able to do this, but are confident your integration works, you may contact productfeed@beslist.nl and request an API key for production. This, however, is at your own risk.

2. Conventions used

How URI's are formed

The Shopitem API is a REST API. Urls are constructed to locate items within a shop. A typical URI within the API should be interpreted like this:

https://shopitem.api.beslist.nl/product/v1/shops/12345/items/1234b45

"item with unique identifier (externalId) 1234b45 within the collection of items of shop with id 12345"

Prices, delivery costs and delivery times are treated as resources themselves in urls like:

https://shopitem.api.beslist.nl/product/v1/shops/12345/items/a123b45/price

Because an item has only one price, there is no unique identifier for it.

Http methods

http methods are used to distinguish between the different types of actions possible. Currently GET and PUT are supported:

- GET requests are used to retrieve data. The following CURL request is a GET request that will retrieve a shopitem:

  ```
  curl -X  GET
  https://shopitem.api.beslist.nl/product/v1/shops/12345/items/1234b45
  ```

- PUT requests are used to update shopitems. The following CURL request is used to update the price of a shopitem:

  ```
  curl -X  PUT
  https://shopitem.api.beslist.nl/product/v2/shops/12345/items/1234b45/
  price
  ```

Currently GET and PUT are the only supported http methods.

Uploading product batches

Not only does the API support single product updates, it is also possible to upload a batch of products with specific updates per product. When the API accepts the batch, a UUID is returned and processing of the product update batch continues in the background.

Through a second endpoint, called with the supplied UUID, you can monitor the status of your batch upload. You will be returned a current upload status or possible failure. Once the upload is complete, each product in the batch will have it's own status code indicating success or failure.

Do realize that you need to make at least two API calls to successfully implement the batch endpoint. One call for uploading the batch of product updates, and one or more calls to confirm the batch processing status.

Please refer to the "Example PUT requests" section for more detailed information about product batches

Responses and http status codes

API responses always return an http status code and can return content. If content is returned, API clients using the API v1 must expect JSON.

The returned http status codes are further documented in the online documentation.

3. Code examples

The following code examples are not to be used in a real environment, but can help as an example of how to consume the Shopitem API.

Getting a list of shops associated with an API key

```
curl -X GET -H "apikey:
12345678abcdef13435373892392902020239О3abcdabc356748494903023abcfde"
https://shopitem.api.beslist.nl/auth/v1/shops
```

This curl request can either return a 401/Unauthorized status code with a json response body if the apikey is invalid, or a 200/OK status code with a list of shops for which the apikey can be used. The list of shops are key-value pairs in json, where the key is the shop_id that must be used in other API calls:

```
{
  "12345": "Debesteshopvoordingen.eu",
  "75634": "Goedkopespullenkopen.nl"  }
```

Two things should be noted:

1. This API endpoint is a bit of an exception, as it does not return a list of shop resources, but a list of key-value pairs.

2. The keys in this list are the shop_id values that must be used in the other endpoints

Listing a shopitem

```
curl –X GET –H "apikey:
        12345678abcdef1343537389239290202023903abcdabc356748494903023abcfde"
        https://shopitem.api.beslist.nl/product/v1/shops/12345/items/12abcd13
```

This curl request will return a 200/OK status code on a successful request and a json representation of the shopitem of “Debesteshopvoordingen.eu” with unique identifier 12abcd13.

```
{
  "price": {
    "last_update": "2015-09-18T16:42:00+0200",
    "value": 21.99
  },
  "discount_price": {
    "last_update": "2015-09-18T16:42:00+0200",
    "value": 19.99
  },
  "delivery_cost_nl": {
    "last_update": "2015-09-18T16:42:00+0200",
    "value": 1.99
  },
  "delivery_cost_be": {
    "last_update": "2015-09-18T16:42:00+0200",
    "value": 3.50
  },
  "delivery_time_nl": {
    "last_update": "2015-09-18T16:42:00+0200",
    "commercial": "Op werkdagen voor 21:00 uur besteld, volgende dag in huis"
},
  "delivery_time_be": {
    "last_update": "2015-09-18T16:42:00+0200",
    "commercial": "Op werkdagen voor 21:00 uur besteld, volgende dag in huis"
},
  "stock": {
    "last_update": "2015-09-18T16:42:00+0200",
    "value": 12
  }
}
```

A 404/Not found is returned when the shopitem cannot be found.

The item_id values used as unique identifiers for the items can differ per shop, but are the same identifiers that are used in the feeds for the shops.

Updating properties for an item

For updates to an item the PUT method is available. Version 1 of the API allows to update single attributes of an item or updates to multiple attributes in one request. For the single-attribute updates there are endpoints for:

- Price (including discount price)

- Delivery time Netherlands
- Delivery time Belgium
- Delivery cost Netherlands*
- Delivery cost Belgium*

The price endpoint gives the opportunity to update a discount price as well as a price for an item. If a discount price is set in the PUT request the price must be set as well. A discount price can be unset with a zero-value.

*Please note that the delivery cost displayed on the beslist website may be affected by your delivery cost settings in the Client Login.

Formats

Prices are formatted as floats, using a decimal point and two decimal digits. Delivery times are strings and are converted to a standardized delivery time strings if recognized as a known string format.

Example PUT requests

The following call will update item 12abcd13 of shop 12345 with a delivery cost for the Netherlands of 3.50 euro:

```
curl -X PUT -H "apikey:
12345678abcdef1343537389239290202023903abcdabc356748494903023abcfde" -d
delivery_cost_nl=3.50  https://shopitem.api.beslist.nl/product/v2/shops/12345/items/12abcd13
```

Updating price and discount price:

```
curl -X PUT -H "apikey:
12345678abcdef1343537389239290202023903abcdabc356748494903023abcfde" -d price=10.99 -d
discount_price=8.49  https://shopitem.api.beslist.nl/product/v2/shops/12345/items/12abcd13
```

Unsetting a discount price:

```
curl -X PUT -H "apikey:
12345678abcdef1343537389239290202023903abcdabc356748494903023abcfde" –d price=10.99 –d
discount_price=0  https://shopitem.api.beslist.nl/product/v2/shops/12345/items/12abcd13
```

Updating stock:

The following call will update the available stock for item 12abcd13 of shop 12345 to 12.

Please note:

- When stock value is smaller than the number of items ordered in the shopping cart, a warning will be displayed during checkout)

- Stock is not updated by beslist when an item is purchased. This has to be updated after a sale using this endpoint of the ShopItemAPI.

```
curl -X PUT -H "apikey:
12345678abcdef134353738923929020202390 3abcdabc356748494903023abcfde" -d
stock=12 https://shopitem.api.beslist.nl/product/v2/shops/12345/items/12abcd13
```

Uploading a product batch:

```
curl -X PUT -H "apikey:

12345678abcdef1343537389239290202023903abcdabc356748494903023abcfde" -d [{ "externalId":
"your-item-identifier", "price": 2.20, "delivery_time_nl": "3 dagen", "delivery_cost_nl":
2.95, "delivery_time_be": "1 week", "delivery_cost_be": 4.95 }, { "externalId":
"youritemidentifier", "price": 13.95 }]
https://shopitem.api.beslist.nl/product/v2/shops/12345/items
```

This first call will upload your batch of updates, and return the exact url to call for monitoring the batch status, including UUID:

```
{
  "url": "https://shopitem.api.beslist.nl/product/v2/shops/101344/batch/8a4a6cda-
42464c68bcc7-54238081dab4"  }
```

The returned url can be called as follows:

```
curl -X GET -H "apikey:
12345678abcdef1343537389239290202023903abcdabc356748494903023abcfde”

https://shopitem.api.beslist.nl/product/v2/shops/12345/batch/8a4a6cda-4246-4c68-bcc7-54238081dab4
```

A possible response when a batch is created, but not yet finished processing:

```
{
  "batch": "8a4a6cda-4246-4c68-bcc7-54238081dab4",
  "status": "new",
  "count": 2,
  "processed": 0,
  "items": []  }
```

When a batch has been completed, the following possible response can be expected:

```
{
  "batch": "8a4a6cda-4246-4c68-bcc7-54238081dab4",
  "status": "finished",
  "count": 2,
  "processed": 2,
  "items": [
    {
      "externalId": "111479",
      "status": 200,
      "response": {
        "status": "success",
        "data": {
          "price": {
            "Value": 100,
            "LastUpdate": "2016-07-29T16:19:12.000000+0200"
          },
          "delivery_time_nl": {
            "Value": {
              "Commercial": "3 werkdagen",
              "ATP": null
            },
            "LastUpdate": "2016-07-29T16:19:12.000000+0200"
          }
        }
      }
    },
    {
      "externalId": "111480",
      "status": 404,
      "response": {
        "status": "error",
        "message": "item could not be found"
      }
    }
  ] }
```

As you can see in the response, the batch was successfully processed, contains 2 products which are both processed, one product was updated and one product could not be found.

Optional request parameters, empty values and 400/Bad Request responses

When a parameter is marked as 'optional' it will be validated if and only if it is provided in the request. It is important to realize that a parameter is considered to be present if the parameter name is present in the request, regardless of the value. When a parameter without a value is passed in the request, validation will almost always fail, making the request return a Http 400/Bad Request status code, because the parameter value will be considered empty, and empty values are almost always invalid.

If you get a lot of 400 responses to requests that look okay at first glance, it may be the case that empty optional parameters are passed.

For convenience, in newer versions of the API the endpoint validations are a bit more relaxed and most of the times optional parameters with empty values will be filtered out. However, this is for convenience and should not be relied upon.

4. Error status codes

Http status codes are used in responses to indicate success and failure. Successful requests get a response in the 2xx range (almost always a 200/OK). Status codes in 4xx and 5xx ranges are returned when a request could not be handled. A status code in the 5xx range is returned whenever a serverside error occurred. A status code in the 4xx range indicates a problem with the request. This usually (but not always) means that the request can be repaired in some way to trigger a successful response. If a response message is returned with a 4xx status code the message usually gives information about the reason of the error response.

The following list describes some scenario's that can trigger error responses in the 4xx range.

HTTP status code	Possible cause
400	Invalid arguments, or in general an invalid request. Is returned when arguments are in an incorrect format (e.g. non-numeric prices)
401	Invalid or missing API key. API keys must be provided as an http header
403	Access is forbidden. This usually means that the client ip address is not allowed access to the API
404	Product cannot be found. This can also happen if a new product was sent in a feed a short time before the API call was done, and the feed was not fully processed yet.
406	Is returned when mandatory arguments are missing

Failing validations also result in HTTP response codes in the 4xx range. For delivery times the following rules apply:

- A delivery time string is looked up and standardized. If the lookup fails, the delivery time is rejected