EXTERNAL API v1.29

DOCUMENTATION

1 About this document

This document lists and explains the API methods available for consumption by third parties, as well as how to authenticate prior to using the API.

2 Version history

May 8, 2019	Filip DUYCK	v1.0	Initial version
July 22, 2020	Filip DUYCK	v1.1	Added customer search, item search
Sept 1, 2020	Filip DUYCK	v1.2	Customer search params: None required, added status Customer search results: Added parent, parameters Article search params: Added status, customer number Article search results: Added memo, remark, parameters, class, type, code, customer-specific prices
Sept 4, 2020	Filip DUYCK	v1.3	Customer creation: Added parent customer number, now checking for existing customer by name/address if no Yerpa or VAT number given. Order creation: Using price from current tender if available. Using parent customer tender if available.
Oct 1, 2020	Filip DUYCK	v1.4	Article information now contains article code (field articleCode). Customer information is updated if desired when sending new order. Ability to include customer status when creating new customer.
Nov 13, 2020	Filip DUYCK	v1.5	Ability to specify tender status to filter customer-specific prices, or leave out customer-specific prices completely, in item search (tenderStatus and includeCustomerPrices parameters) – see 7.1. Endpoints to retrieve tender information – see 9.1, 9.2.
Nov 18, 2020	Filip DUYCK	v1.6	Added endpoints to get lists of item codes, classes and types – see 7.4 and beyond.

Nov 20, 2020	Filip Duyck	v1.7	Ability to create new customers (see 8.1) and new customer sites (see 8.2). Customer information now includes customer sites (see 8.3). Ability to specify customer site when creating new sale orders (see 10.1).
Dec 11, 2020	Filip DUYCK	v1.8	Added ability to include price when placing an order (see 10.1).
Dec 24, 2020	Filip DUYCK	v1.9	When creating a sale order, remark and customer reference can be passed along (see 10.1).
Jan 12, 2021	Filip DUYCK	v1.10	Article information now contains default quantity.
Jan 14, 2021	Filip DUYCK	v1.11	Customer site information now contains email and phone number.
May 11, 2021	Filip DUYCK	v1.12	When creating new customer/site records and postal code is not found, a new place entry will be created in Yerpa. Now accepting “country” parameter when creating new customers/sites. Renamed “countryCode” to “country” and “cityName” to “city” for several API methods, but providing backward compatibility.
Aug 13, 2021	Filip DUYCK	v1.13	Added ability to attach a person to an order by number (see 10.1).
Jan 21, 2022	Filip DUYCK	v1.14	Customer prices now also include discount information (see 7.1.2 and 9.2.2). Added information about token-based authentication mechanism (see 5.1).
Jul 20, 2022	Filip DUYCK	v1.15	Added sale invoice and purchase delivery methods (see 11.1 and 12.1).
Jul 24, 2022	Filip DUYCK	v1.16	Added ability to pass properties/parameters when creating a sale order (see 10.1).
Aug 25, 2022	Filip DUYCK	v1.17	Added documentation on how to send properties/parameters (see 13).
Sep 2, 2022	Filip DUYCK	v1.18	Added information on creating and updating new items (see 7.2 and 7.3).
Oct 19, 2022	Filip DUYCK	v1.19	Added ability to pass unitCode when adding/updating new items (see 7.2.1 and 7.3.1). Added documentation for searching and retrieving sale orders (see 10.2 and 10.3). Added parameter condition when searching for customers (see 8.3.1).
Dec 15, 2022	Filip DUYCK	v1.20	Added methods to list, add and delete popups for existing items (see 7.7 and beyond).
March 10, 2023	Filip DUYCK	v1.21	Now hiding archived items by default and including a ShowArchived request field on relevant endpoints.
Feb 21, 2024	Filip DUYCK	v1.22	Added documentation about “city” field to sale order creation method (see 10.1).
Mar 8, 2024	Filip DUYCK	v1.23	Added simple paging to item search method.

Mar 27, 2024	Filip DUYCK	v1.24	Added “label” and “status” fields to item search and retrieval methods. Added additional status filtering options to item search method. (See 7.1)
Apr 10, 2024	Filip DUYCK	v1.25	Deprecated “codeNumber”, “classNumber” and “typeNumber” fields in favor of “code”, “class” and “type” which contain objects with fields “id” and “number” (see 7). Added optional startDate/endDate fields when creating sale order (see 10.1).
Apr 18, 2024	Filip DUYCK	v1.26	Added code/type/class information for customers (currently only during retrieval - see 8).
Apr 24, 2024	Filip DUYCK	v1.27	Added status fields to order and orderline objects when retrieving a sale order’s details (see 10.3.2).
Apr 3, 2025	Filip DUYCK	v1.28	Including modification dates for sale order, sale order lines and sale order line parameters. Including barcode for items. Added calls to retrieve sale order line details and update quantity (see 10.4 and 0). Added “extraItemFields” parameter for sale order / sale order item retrieval.
May 8, 2025	Filip DUYCK	v1.29	Added methods for adding and removing sale order lines (see 10.5 and 10.7).

3 Table of contents

4 General information

4.1 API host

The API host to communicate with is on the same URL as your regular Yerpa website. If you use https://company.yerpa.eu to log in to Yerpa, your API calls should be directed towards https://company.yerpa.eu/api .

4.2 Communication format

Unless explicitly specified otherwise, all API methods:

- expect to be called using the HTTP `POST` method,
- expect to receive input as a JSON-formatted request body (`application/json`), and
- return a JSON-formatted response body.

5 Authentication

With the exception of the authentication methods themselves, all API methods require the client to include authentication information, either using a fixed token or a hash referring to an authenticated session.

5.1 Authentication with fixed token

This is our preferred way of authentication for third parties. You will have received a permanent but revokable API token that can be used to talk to the Yerpa API.

You can pass this token along with your request as a HTTP header named `X-API-Token`.

Note: Many of the examples in this document will still have a `loginHash` field in the request body. This field is for session based authentication and should be left out when using the `X-API-Token` header.

5.2 Authentication with session hash

This authentication method is considered legacy and should only be used in absence of an API token. The API caller is expected to open a session with Yerpa prior to calling other API methods. This can be done using the `authenticate` method, as follows:

```
POST /api/v1/authenticate
{
    "username": "myuser@mycompany.com",
    "password": "mypassword"
}
```

Yerpa will respond with a token that you can use in your future calls:

```
{
    "loginHash": "c1c1d41fe2f24ce62a34b"
}
```

Any calls to other API methods should include this token in one of three ways: as a query string parameter:

```
/api/v1/ping?loginHash=c1c1d41fe2f24ce62a34b
```

or as a JSON body field:

```
POST /api/v1/ping
{
    "loginHash": "c1c1d41fe2f24ce62a34b",
    ...
}
```

as a HTTP header:

```
POST /api/v1/ping
Content-Type: application/json
LoginHash: c1c1d41fe2f24ce62a34b
...
{
    ...
}
```

6 Testing authentication

As you might have noticed, there is a simple method `/api/v1/ping` that you can call during development to test if your authentication is working correctly. It will always respond with the simple string "Pong."

```
POST /api/v1/ping
{
    "loginHash": "c1c1d41fe2f24ce62a34b"
}

→

"Pong."
```

7 Item methods

7.1 Search for items

This method allows the caller to search for one or more items from the catalog.

```
POST /api/v1/items/search
{
    "loginHash": "c1c1d41fe2f24ce62a34b",
    "classNumber": 1,
    "status": "webshop",
    "tenderStatus": "webshop",
    "includeCustomerPrices": false,
    "showArchived": true,
    "pageSize": 1000,
    "currentPage": 3,
    "status": "webshop,!inactive"
}

→

{
    "items": [
        {
            "number": "CI/1.102",
            "name": "Article 102",
            "articleCode": "12345"
            "barcode": "500040330304022",
            "description": "Article with a description",
            "remark": "Item with a remark",
            "memo": "<p>Item with a memo</p>",
            "code": { "id": 223, "number": "1.1" },
            "class": { "id": 31, "number": "2" },
            "type": { "id": 1882, "number": "3.2" },
            "unitCode": "stuk",
            "purchasePrice": 17.9000,
            "salePrice": 39.380000,
            "defaultVatName": "T40 21%",
            "defaultVatPercentage": 21.00,
            "defaultVat": 8.26980000,
            "salePriceIncludingDefaultVat": 47.64980000,
            "label": "This is the label",
            "status": [ "Webshop", "OnHold" ]
        },
        {
            "number": "CI/1.107",
            "name": "Article 107",
            "purchasePrice": 52.9000,
            "salePrice": 116.380000,
```

```
            "defaultAmount": 15,
            "defaultVatName": "T40 21%",
            "defaultVatPercentage": 21.00,
            "defaultVat": 24.43980000,
            "salePriceIncludingDefaultVat": 140.81980000,
            "label": "This is the label",
            "status": [ "Webshop" ],
            "properties": [
                {
                    "name": "Foto",
                    "parameters": [
                        {
                            "name": "Foto",
                            "value": "6510_1120_011.jpg"
                        }
                    ]
                }
            ],
            "customerPrices": [
                {
                    "customer": "C/12712",
                    "unitSalePrice": 30.525000,
                    "unitSalePriceIncludingVat": 36.93525000,
                    "vatName": "T40 21%",
                    "vatPercentage": 21.00,
                    "discount": 10.00
                },
                …
            ]
        }
    ]
}
```


7.1.1 Request

One or more of the following parameters can be passed to the API method to refine the search:

- `number`: (optional) Only return the item with the given number (eg.: CI/12712.133).
- `classNumber`: (optional) Number of the item class to which this item is attached (eg.: 1).
- `status`: (optional) Only show items that have this status set (eg.: webshop).
- `tenderStatus`: (optional) When displaying customer-specific prices, only use tenders that have this status set.
- `customerNumber`: (optional) Only show items that have a price specific to the given customer. Omits price information for all other customers (eg.: C/12712).
- `includeCustomerPrices`: (optional) Whether or not to include customer-specific prices in the response. Defaults to true.
- `showArchived`: (optional) Whether or not to include archived items in the response. Defaults to false.

- `pageSize`: (optional) Page size between 1 and 1000. When omitted, all results will be returned (which might or might not be a good idea).
- `currentPage`: (optional) Current page (starting at 1). Ignored if `pageSize` is omitted.
- `status`: (optional) Formula specifying which status the items should or should not have in order to be included in the list. See the chapter on status filtering for more information (14).

7.1.2 Response

The response will contain an array called `items` that contains all the items corresponding to the given search query.

For every entry, `number`, `name` and `status` will always be included. `articleCode`, `barcode` and `unitCode` will be included if specified.

The fields `description`, `memo`, `label` and `remark` might be included, or might be omitted if no unit code, description, memo or remark was given for the item. Description and remark are plain text, memo might (but doesn't have to) contain HTML markup.

If available, `defaultAmount`, `purchasePrice` and `salePrice` will be provided. If `salePrice` is available, a default VAT calculation will also be made. However, the actual VAT calculation for a given order might differ from this default value, if a different VAT regime Is specified for the customer who made the order.

The fields `code`, `class` and `type` will contain the IDs and numbers (as an object with `id` and `number` fields) of the assigned code, class and/or type for this item, if available.

The fields ~~`codeNumber`~~, ~~`classNumber`~~ and ~~`typeNumber`~~ will contain the numbers of the assigned code, class and/or type for this item, if available. **These fields are still present but deprecated and will be removed in a future version.**

The `properties` field will contain an array of properties, each of which have an array of parameters in their `parameters` field.

Customer-specific prices, if requested, will be listed in the `customerPrices` array. For each customer, the customer number (`number`) and unit sale price (`unitSalePrice`) will be included, as well as the used VAT regime (`vatName` & `vatPercentage`) and a calculation of the unit price including VAT (`unitSalePriceIncludingVat`). Optionally a `discount` field might be present containing an extra discount percentage for this customer. The discount has **not** yet been applied to the listed price.

7.2 Creating a new item

This API method can be used to create a new item from scratch.

```
POST /api/v1/items
{
    "companyNumber": "C/4426"
    "name": "Testartikel",
    "articleCode": "4523-93211-348",
```

```
    "description": "Een artikel om te testen",
    "remark": "geen",
    "memo": "geen",
    "codeNumber": "1",
    "typeNumber": "2",
    "classNumber": "1.3",
    "vatNumber": "CT/1.4",
    "defaultAmount": 1,
    "purchasePrice": 20,
    "salePrice": 30,
    "unitCode": "kg"
}

→

{
    "item": {
        "number": "CI/4426.7",
        "name": "Testartikel",
        "articleCode": "4523-93211-348",
        "unitCode": "kg",
        "description": "Een artikel om te testen",
        "remark": "geen",
        "memo": "geen",
        "code": { "id": 223, "number": "1.1" },
        "class": { "id": 31, "number": "2" },
        "type": { "id": 1882, "number": "3.2" },
        "defaultAmount": 1.00,
        "purchasePrice": 20.0000,
        "salePrice": 30.000000,
        "defaultVatName": "T40 21%",
        "defaultVatPercentage": 21.00,
        "defaultVat": 6.30000000,
        "salePriceIncludingDefaultVat": 36.30000000,
        "properties": []
    }
}
```

7.2.1 Request

When creating a new item, the following fields are supported. All of them are optional.

companyNumber	Yerpa number of the company this item should be linked to.
name	Name of this item
articleCode	Article code of this item
description	Description of this item
remark	Remark for this item
memo	Memo for this item
codeNumber	Yerpa number of the “code” classifier linked to this item
typeNumber	Yerpa number of the “type” classifier linked to this item

classNumber	Yerpa number of the “class” classifier linked to this item
vatNumber	Yerpa number of the VAT regime this item will use by default
defaultAmount	Default number of items to add when adding to an order
purchasePrice	Purchase price, VAT not included
salePrice	Sale price, VAT not included
unitCode	The unit code to use for this item, as specified in Yerpa’s Configuration -> Home -> General -> Units list.

7.2.2 Response

The response will contain the newly created item, using the same format as described in 7.1.2

7.3 Updating an existing item

This API method can be used to update an existing item identified by its Yerpa number.

```
POST /api/v1/items/4426.7
{
    "purchasePrice": 20,
    "salePrice": 30,
}

→

{
    "item": {
        "number": "CI/4426.7",
        "name": "Testartikel",
        "articleCode": "4523-93211-348",
        "description": "Een artikel om te testen",
        "remark": "geen",
        "memo": "geen",
        "unitCode": "kg",
        "code": { "id": 223, "number": "1.1" },
        "class": { "id": 31, "number": "2" },
        "type": { "id": 1882, "number": "3.2" },
        "defaultAmount": 1.00,
        "purchasePrice": 20.0000,
        "salePrice": 30.000000,
        "defaultVatName": "T40 21%",
        "defaultVatPercentage": 21.00,
        "defaultVat": 6.30000000,
        "salePriceIncludingDefaultVat": 36.30000000,
        "properties": []
    }
}
```

7.3.1 Request

The Yerpa number of the item to be updated should be mentioned in the URL. An item with number CI/4426.7 is updated by performing a POST request to /api/v1/items/4426.7.

When updating an existing item, the following fields are supported. All of them are optional.

name	Name of this item
articleCode	Article code of this item
description	Description of this item
remark	Remark for this item
memo	Memo for this item
codeNumber	Yerpa number of the “code” classifier linked to this item
typeNumber	Yerpa number of the “type” classifier linked to this item
classNumber	Yerpa number of the “class” classifier linked to this item
vatNumber	Yerpa number of the VAT regime this item will use by default
defaultAmount	Default number of items to add when adding to an order
purchasePrice	Purchase price, VAT not included
salePrice	Sale price, VAT not included
unitCode	The unit code to use for this item, as specified in Yerpa’s Configuration -> Home -> General -> Units list.

7.3.2 Response

The response will contain updated item, using the same format as described in 7.1.2

7.4 Item codes

Returns a list of available item codes.

```
POST /api/v1/items/codes
{
    "loginHash": "c1c1d41fe2f24ce62a34b"
}

→

{
    "codes": [
        {
            "id": 14,
            "number": "1",
            "name": "An item code"
        },
        {
            "id": 15,
            "number": "1.1",
            "parentNumber": "1",
```

```
            "name": "A child item code"
        },
        …
    ]
}
```

7.5 Item types

Returns a list of available item types.

```
POST /api/v1/items/types
{
    "loginHash": "c1c1d41fe2f24ce62a34b"
}

→

{
    "types": [
        {
            "id": 24,
            "number": "1",
            "name": "An item type"
        },
        {
            "id": 28,
            "number": "1.1",
            "parentNumber": "1",
            "name": "A child item type"
        },
        …
    ]
}
```

7.6 Item classes

Returns a list of available item classes.

```
POST /api/v1/items/classes
{
    "loginHash": "c1c1d41fe2f24ce62a34b"
}

→

{
    "classes": [
        {
            "id": 71,
            "number": "1",
```

```
            "name": "An item class"
        },
        {
            "id": 73,
            "number": "1.1",
            "parentNumber": "1",
            "name": "A child item class"
        },
        …
    ]
}
```

7.7 Listing popups for an item

Popups are special notifications that appear in Yerpa when certain documents are opened. In the context of items, these popups will appear when viewing the item or orders/deliveries/invoices/... that contain this item.

```
GET /api/v1/items/1543.324/popups

→

{
    "popups": [
        {
            "number": "CA/22.2253",
            "name": "Korting",
            "description": "Korting geldig van 22 tot 25 december"
        },
        …
    ]
}
```

The number of the item for which we want to list the popups is passed in through the URL (minus the CI/ prefix).

Returned popups will include `number`, `name` and `description` of the popup. A popup's name is informational – only the description is shown in the actual popup!

7.8 Creating a new popup for an item

You can create multiple popups per item, one at a time. Technically speaking, both `name` and `description` are optional, but evidentally the value of a popup lies in its message. In other words, while you can leave `name` empty, it is strongly advised to enter a value for `description` as this is the message that will be shown in Yerpa.

```
POST /api/v1/items/1543.324/popups
{
    "name": "Korting",
    "description": "Korting geldig van 22 tot 25 december”
}

→

{
    "success": true,
    "createdPopup": {
        "number": "CA/22.2253",
        "name": "Korting",
        "description": "Korting geldig van 22 tot 25 december"
    },
    "popups": [
```

```
        {
            "number": "CA/22.2252",
            "name": "Korting",
            "description": "Korting van 20 tot 23 juni"
        },
        {
            "number": "CA/22.2253",
            "name": "Korting",
            "description": "Korting van 22 tot 25 december"
        }
    ]
}
```

7.9 Removing an existing popup for an item

It is possible to delete a single popup by specifying the item number and popup number in the URL during a delete call:

```
POST /api/v1/items/1543.324/popups/22.2252/delete
→

{
    "success": true,
}
```

8 Customer methods

8.1 Create new customers

This method allows the caller to create a new customer in the Yerpa database. If a customer with the same VAT number already exists, this customer will not be created anew, and the existing customer information will be returned.

```
POST /api/v1/customers/create
{
    "name": "The Company bvba",
    "vat": "BE0664359968",
    "address": "Dorpstraat 1",
    "postalCode": "BE 9000",
    "city": "Gent",
    "phone": "+32-473-123456",
    "email": "manager@thecompany.be",
    "language": "nl",
    "country": "BE"
}

→
```

```
{
    "number": "C/13177",
    "companyName": "The Company bvba",
    "vat": "BE0664359968",
    "address": "Dorpstraat 1",
    "postalCode": "BE 9000",
    "city": "Gent",
    "country": "BE",
    "phone": "+32-473-123456",
    "email": "manager@thecompany.be",
    "sites": []
}
```

8.1.1 Request

parentCompanyNumber	The number of the parent company (usually not necessary)
companyName	The name of the company
vat	The VAT number for this company, formatted <country code><number>, e.g. "BE0665448694". For customers without a VAT number, specify a single dash ("-").
address	Street address, eg. "Dorpstraat 4 bus 5"
postalCode	Postal code. This identifies both the city and country. The passed in value should correspond to the "Code" value of this place within Yerpa, eg. "BE 1000".
city	Name of the city. This is important when the postal code is not found and a new place needs to be created in the system, or for postal codes that apply to several places.
country	2-character ISO country code (eg. BE, DE, …) (optional)
phone	Contact phone number
name	Contact person full name
email	Contact email address
language	ISO 2 letter code
statuses	Array of statuses to set on the customer (if new).

8.1.2 Response

The information for the created or existing customer will be returned in the response. The response will also include a `number` field that contains the Yerpa number for the created or existing customer.

8.2 Create new customer site

This method allows the caller to create new so called customer sites: extra addresses that can be linked to a customer and can act as (among others) delivery addresses if an order needs to be shipped to a place other than the invoicing address.

If the address matches another site for the same customer, no new site will be created and the information for the existing site will be returned instead.

```
POST /api/v1/customers/sites/create
{
    "customerNumber": "C/13177",
```

```
    "name": "Site name",
    "address": "Dorpstraat 2",
    "postalCode": "BE 9000",
    "city": "Gent",
    "country": "BE"
}

→

{
    "number": "C/13177.1",
    "name": "Site name",
    "address": "Dorpstraat 1",
    "postalCode": "BE 9000",
    "city": "Gent",
    "country": "BE"
}
```

8.2.1 Request

Your request needs to include the `customerNumber` of the customer for which you are adding the site.

You should also include `name`, `address`, `postalCode` and `city` field for the site to be created. The `postalCode` refers to the "place code" in Yerpa, so depending on your Yerpa setup, you might need to add "BE" in front of your regular postal code.

Optionally you can also pass in `country`, `email` and `phone` fields.

8.2.2 Response

The response will repeat the information you passed in, as well as return the Yerpa number of the created site.

8.3 Search for customers

This method allows the caller to search for one or more customers already existing in the Yerpa database, based on a combination of known fields.

```
POST /api/v1/customers/search
{
    "loginHash": "c1c1d41fe2f24ce62a34b",
    "number": "C/1",
    "companyName": "The Company BV",
    "address": "Stationsplein 5",
    "postalCode": "9000",
    "cityName": "Gent",
    "country": "BE",
    "status": "webshop",
    "parameter": {
        "name": "exact online code",
        "value": 1805
    },
```

```
    "showArchived": false,
}

→

{
    "customers": [
        {
            "number": "C/12",
            "parentCompanyNumber": "C/3",
            "companyName": "Bedrijf met BTWnummer",
            "vat": "BE043275264",
            "name": null,
            "address": "Stationsplein 5",
            "postalCode": "BE 9000",
            "city": "Gent",
            "country": "BE",
            "phone": "09/225 99 10",
            "email": "info@bedrijf.be",
            "code": { "id": 223, "number": "1.1" },
            "class": { "id": 31, "number": "2" },
            "type": { "id": 1882, "number": "3.2" },
            "properties": [
                {
                    "name": "Informatie",
                    "parameters": [
                        {
                            "name": "Informatie",
                            "value": "123"
                        }
                    ]
                }
            ]
            "sites": [
                {
                    "number": "CD/12.1",
                    "name": "Filiaal 2",
                    "address": "De Pintelaan 24",
                    "postalCode": "BE 9000",
                    "city": "Gent",
                    "country": "BE",
                    "phone": "09/6653223",
                    "email": "filiaal2@bedrijf.be"
                },
                …
            ]
        }
    ]
}
```

8.3.1 Request

Several parameters can be passed to the API method to refine the search. All of them are optional.

number	The Yerpa number of the customer
companyName	The name of the customer
vat	The VAT number for this customer, formatted <country code><number>, e.g. "BE0665448694".
address	Street address, eg. "Dorpstraat 4 bus 5"
postalCode	Postal code. This identifies both the city and country. The passed in value should correspond to the "Code" value of this place within Yerpa, eg. "BE 1000".
city	Name of the city.
country	Country code as specified in Yerpa.
status	Name of a status that needs to be set on the customers (eg. webshop)
parameter	An object containing a parameter `name` and `value` that needs to be present on the customer records.
showArchived	Whether or not to show archived companies. Defaults to false.

8.3.2 Response

The response will contain an array called `customers` that contains all the items corresponding to the given search query.

For every entry, `number` and `companyName` will always be included. All other fields are optional.

The fields `code`, `class` and `type` will contain the IDs and numbers (as an object with `id` and `number` fields) of the assigned code, class and/or type for this customer, if available.

The `properties` field will contain an array of properties, each of which have an array of parameters in their `parameters` field.

If available, company sites will be specified in the `sites` field. Sites have `number`, `name`, `address`, `phone` and `email` fields.

8.4 Customer codes

Returns a list of available customer codes.

```
POST /api/v1/customers/codes
{
    "loginHash": "c1c1d41fe2f24ce62a34b"
}

→

{
    "codes": [
        {
            "id": 14,
```

```
            "number": "1",
            "name": "A customercode"
        },
        {
            "id": 15,
            "number": "1.1",
            "parentNumber": "1",
            "name": "A child customer code"
        },
        …
    ]
}
```

8.5 Customer types

Returns a list of available customer types.

```
POST /api/v1/customers/types
{
    "loginHash": "c1c1d41fe2f24ce62a34b"
}

→

{
    "types": [
        {
            "id": 24,
            "number": "1",
            "name": "A customer type"
        },
        {
            "id": 28,
            "number": "1.1",
            "parentNumber": "1",
            "name": "A child customer type"
        },
        …
    ]
}
```

8.6 Customer classes

Returns a list of available customer classes.

```
POST /api/v1/customers/classes
{
    "loginHash": "c1c1d41fe2f24ce62a34b"
}
```

→

```
{
    "classes": [
        {
            "id": 71,
            "number": "1",
            "name": "An customers class"
        },
        {
            "id": 73,
            "number": "1.1",
            "parentNumber": "1",
            "name": "A child customers class"
        },
        …
    ]
}
```

9 Sale tender methods

9.1 Search for tenders

This method allows the caller to search for one or more sale tenders based on a combination of known fields.

```
POST /api/v1/sale/tenders/search
{
    "loginHash": "c1c1d41fe2f24ce62a34b",
    "customerNumber": "C/1",
    "status": "webshop",
    "showArchived": true
}
```

→

```
{
    "tenders": [
        {
            "number": "ST/19.362",
            "customerNumber": "C/1"
        },
        {
            "number": "ST/19.158",
            "customerNumber": "C/1"
        },
        {
```

```
            "number": "ST/19.271",
            "customerNumber": "C/1"
        },
        …
    ]
}
```

9.1.1 Request

Several parameters can be passed to the API method to refine the search. All of them are optional.

customerNumber	The Yerpa number of the customer
status	Name of a status that needs to be set on the tenders (eg. webshop)
showArchived	Whether or not to show archived tender. Defaults to false.

9.1.2 Response

The response will contain an array called `tenders` that contains all the tenders corresponding to the given search query. For every tender, the tender `number` and `customerNumber` will be included.

This tender number can then be used to get the tender details with the method explained below.

9.2 Retrieve single tender

This method allows the caller to search for one or more sale tenders based on a combination of known fields.

```
POST /api/v1/sale/tenders/20.425
{
    "loginHash": "c1c1d41fe2f24ce62a34b",
}

→

{
    "tender": {
        "number": "ST/20.425",
        "customerNumber": "C/13174",
        "startDate": "2020-11-10T00:00:00",
        "items": [
            {
                "number": "CI/11258.14789",
                "name": "Item name",
                "articleCode": "Article code",
                …,
                "customerPrices": [
                    {
                        "customer": "C/13174",
                        "unitSalePrice": 55.850000,
```

```
                        "unitSalePriceIncludingVat": 67.57850000,
                        "vatName": "T40 21%",
                        "vatPercentage": 21.00,
                        "discount": 10.00
                    }
                ]
            },
            …
        ]
    }
}
```

9.2.1 Request

The tender number (minus the ST/ prefix) is passed in as part of the URL. There are no request body parameters (except for the `loginHash` when not sending this as a HTTP header).

9.2.2 Response

The response will contain an object called `tender` that contains the `number`, `customerNumber` and `startDate` for the requested tenders, as well as an `items` array containing the different items listed in the requested tender. Every item will have the same information as the items in the item search (see 7.1.2) with the exception that only the custom price for the customer for which this tender is applicable is included.

10 Sale order methods

10.1 Create sale order

This method creates a new sale order (and perhaps customer) in Yerpa.

10.1.1 Request

```
POST /api/v1/sale/orders/create
{
    "loginHash": "c1c1d41fe2f24ce62a34b",
    "customer": {
        "number": "C/353",
        "vat": "BE0665488896",
        "companyName": "Company A",
        "address": "Stationstraat 5",
        "postalCode": "BE 2000",
        "city": "Antwerpen",
        "name": "Jan Peeters",
        "phone": "+32-3-6001234",
        "email": "jan.peeters@companya.be",
        "language": "NL",
        "statuses": ["webshop"]
    },
```

```
    "customerSite": {
        "number": "CD/353.1"
    },
    "startDate": "2024-04-01",
    "endDate": "2024-04-10",
    "person": "CC/1.18",
    "orderLines": [
        {
            "article": "55453",
            "quantity": 5.5,
            "salePrice": 12.50,
            "properties": [{
                "name": "Handling",
                "parameters": [
                    { "name": "Borduren", "value": "True" },
                    { "name": "Bedrukken", "value": "True" },
                    { "name": "Foto", "value": <base64 encoded file content>,
filename: "myfile.jpg" }
                ]
            }]
        },
        ...
    ]
}

→

{
    "customer": {
        "number": "C/353",
        "companyName": "Company A",
        "vat": "BE0665488896",
        "name": "Jan Peeters",
        "address": "Stationstraat 5",
        "postalCode": "BE 2000",
        "city": "Antwerpen",
        "country": "BE",
        "phone": "+32-3-6001234",
        "email": "jan.peeters@companya.be",
        "sites": [
            {
                "number": "CD/353.1",
                "name": "Filiaal 2",
                "address": "De Pintelaan 24",
                "postalCode": "BE 9000",
                "city": "Gent",
                "country": "BE"
            }
        ]
    },
```

```
    "orderNumber": "SO/19.23",
    "customerSite": {
        "number": "CD/353.1",
        "name": "Filiaal 2",
        "address": "De Pintelaan 24",
        "postalCode": "BE 9000",
        "city": "Gent",
        "country": "BE",
        "phone": "09/6653223",
        "email": "filiaal2@bedrijf.be"
    },
    "remark": "Order related remark",
    "referenceCustomer": "Customer reference"
}
```

We identify a number of parameters that need to be sent to this method:

- `customer`: defines a new or existing customer.
  When creating a new customer, refer to the table given below for more information about the individual properties. When referring to an existing customer, you only need to specify either the `number` (= Yerpa ID for this customer) or `vat` field (assuming the customer has a VAT number).
  When not passing in a Yerpa ID or VAT number BUT specifying a parent company number, Yerpa will look for an existing customer in the database with the same parent company, name, address and postal code.

- `customerSite`: optionally defines a new or existing customer site, used as a delivery address. When creating a new customer site, you should include `name`, `address`, `postalCode` and `cityName` field for the site to be created. The postalCode refers to the "place code" in Yerpa, so depending on your Yerpa setup, you might need to add "BE" in front of your regular postal code. When referring to an existing site, you only need to specify the site number using the `number` field. Optionally you can also pass in `email` and `phone` fields.

- `person`: optionally specifies a person by number to be attached to the order.

- `startDate` / `endDate`: optionally specifies the start- and enddate of a sale order.

- `orderLines`: defines the content of this order

- `remark` and `referenceCustomer` let you pass a remark and customer reference along with the order.

Customer properties:

parentCompanyNumber	The number of the parent company (usually not necessary)
companyName	The name of the company
vat	The VAT number for this company, formatted <country code><number>, e.g.

	"BE0665448694". For customers without a VAT number, specify a single dash ("-").
address	Street address, eg. "Dorpstraat 4 bus 5"
postalCode	Postal code. This identifies both the city and country. The passed in value should correspond to the "Code" value of this place within Yerpa, eg. "BE 1000".
city	Name of the city. This is important when the postal code is not found and a new place needs to be created in the system, or for postal codes that apply to several places.
phone	Contact phone number
name	Contact person full name
email	Contact email address
language	ISO 2 letter code
statuses	Array of statuses to set on the customer (if new).
updateIfExisting	If set to true, existing customer record will be updated with specified information.

If the VAT number given corresponds to an existing customer, the other supplied properties are discarded and the existing customer will be used. If the `updateIfExisting` field is set to true, the existing customer record will be updated with the supplied information (except for `language`, `status`, `number` and `parentCompanyNumber` – these fields can not be updated)

Order line properties:

article	Article code for the given order line. This refers to the customer defined identification number for a given article ("artikel nummer"), not its Yerpa number.
quantity	Number of items to order
salePrice	Unit price (excluding VAT) of the item being ordered (optional – if omitted, Yerpa will look at the currently valid tender)
properties	Allows the caller to set a number of properties on the sale order item.
remark	A remark for the given order line (optional).

Please refer to chapter 13 (Properties and parameters) for more information on sending properties and parameters.

When creating a new order for a customer who has an active tender (meaning the most recent, currently valid tender), prices from this tender will be used if no `salePrice` is specified. If a parent company number was specified and said parent company has a tender, that tender will be used instead.

10.1.2 Response

The response will contain information on the newly created or existing customer, including his Yerpa number. It is advisable to store this number and reuse it later when creating new orders for the same customer.

Apart from the customer information, the response will also include an `orderNumber` field that contains the Yerpa number for this order. This could be interesting information to store, since it allows you to easily locate your order in Yerpa later on.

If the request specified a `customerSite` to which the order should be delivered, the information about this site will be returned in the response's `customerSite` field.

10.2 Search sale orders

This method allows the caller to search for orders based on their customer number or status.

```
POST /api/v1/sale/orders/search
{
    "customerNumber": "C/1",
    "status": "onhold",
    "showArchived": false,
}

→

{
    "orders": [
        {
            "number": "SO/22.362",
            "customerNumber": "C/1",
            "creationDate": "2024-01-25T16:15:19.447",
            "modificationDate": "2025-04-03T11:30:29.757"
        },
        {
            "number": "SO/22.158",
            "customerNumber": "C/1",
            "creationDate": "2024-01-25T16:15:19.447",
            "modificationDate": "2025-04-03T11:30:29.757"
        },
        {
            "number": "SO/22.271",
            "customerNumber": "C/1",
            "creationDate": "2024-01-25T16:15:19.447",
            "modificationDate": "2025-04-03T11:30:29.757"
        },
        …
    ]
}
```

10.2.1 Request

Several parameters can be passed to the API method to refine the search. All of them are optional.

customerNumber	The Yerpa number of the customer
status	Name of a status that needs to be set on the orders (eg. onhold)
handlingParametersUpdatedSince	Only show items whose handling parameters have been updated

	since the specified date.
showArchived	Whether or not to show archived orders. Defaults to false.

10.2.2 Response

The response will contain an array called `orders` that contains all the orders corresponding to the given search query. For every order, the order `number` and `customerNumber` will be included, as well as their `creationDate` and `modificationDate`. The `modificationDate` is not just the time the order itself was modified, but the most recent modification date for the order, order lines or order line parameters.

This order number can then be used to get the order details with the method explained below.

10.3 Retrieve single order

This method allows the caller to retrieve a single order based on its Yerpa number.

```
POST /api/v1/sale/orders/22.425
{
    "loginHash": "c1c1d41fe2f24ce62a34b",
    "extraItemFields": [ "field1", "field2" ]
}

→

{
    "order": {
        "number": "SO/22.425",
        "customerNumber": "C/13174",
        "creationDate": "2024-01-25T16:15:19.447",
        "modificationDate": "2025-04-03T11:30:29.757"
        "startDate": "2022-11-10T00:00:00",
        "status": ["SentByFax", "ToPurchaseOrder"],
        "items": [
            {
                "number": "SO/22.425.1",
                "itemNumber": "CI/11258.14789",
                "name": "Item name",
                "quantity": 1.00,
                "name": "Item name",
                "unitCode": "kg",
                "salePrice": 17.8660,
                "vatPercentage": 21.00,
                "vatName": "T40 21%",
                "status": ["ToPurchaseOrder"],
                "creationDate": "2024-01-25T16:15:19.447",
                "modificationDate": "2025-04-03T11:30:29.757"
                "extraFields": {
```

```
                    "field1 ": 1.0
                },
                "properties": [
                    {
                        "name": "Handling",
                        "parameters": [
                            {
                                "name": "Bedrukken",
                                "value": "True",
                                "creationDate": "2024-01-25T16:15:19.447",
                                "modificationDate": "2025-04-03T11:30:29.757"
                            }
                        ]
                    }
                ]
            },
            …
        ]
    }
}
```

10.3.1 Request

The order number (minus the SO/ prefix) is passed in as part of the URL. There is one parameter that can optionally be specified: `extraItemFields`. This contains an array of extra fields to be retrieved. Consult your Yerpa contact for more information, as only so called “SQL Fields” can be included.

In case of a POST call, this should be a string array, for example:

```
POST /api/v1/sale/orders/22.425
{
    "extraItemFields": [ "field1", "field2" ]
}
```

In the case of a GET call, where no body is specified, it’s also possible to pass the array using querystring parameters, for example:

```
GET /api/v1/sale/orders/22.425?extraItemFields=field1&extraItemFields=field2
```

10.3.2 Response

The response will contain an object called `order` that contains the `number`, `customerNumber`, `creationDate`, `modificationDate` and `startDate` for the requested order, as well as an `items` array containing the different items listed in the requested tender. Unlike the order search function, the `modificationDate` presented on the order level here is applicable only to the order itself, not its order lines or parameters (who both have their own `creationDate` and `modificationDate` fields in this response). There will also be a `status` field which contains an array of strings representing the statuses for this order. The array will be empty if no status was applied to the order.

Every item will have fields for `number`, `name`, `itemNumber`, `quantity`, `unitCode`, `salePrice`, `creationDate`, `modificationDate`, `vatPercentage` and `vatName`. There will also be a field called

`properties` that contains an array of properties, each of which have an array of parameters in their `parameters` field. Similar to the order object itself, each line will also have a `status` field containing a string array. If the `extraItemFields` parameter was specified in the request, every item will also come with an `extraFields` object listing retrieved values for those fields, if present.

10.4 Retrieve single order line

This method allows the caller to retrieve the details of a given sale order line based on its number.

```
GET /api/v1/sale/orders/24.642/items/1

→

{
  "orderLine": {
    "number": "SO/24.642.1",
    "name": "KASSIE | O1 | Safety Jogger | Zwart | 39",
    "itemNumber": "CI/11259.3345",
    "quantity": 5.0,
    "unitCode": "Paar",
    "salePrice": 43.23,
    "properties": [
      {
        "name": "Handling",
        "parameters": [
          {
            "name": "Bedrukken",
            "value": "True",
            "creationDate": "2025-04-03T10:12:06.883",
            "modificationDate": "2025-04-03T10:12:09.23"
          }
        ]
      }
    ],
    "vatPercentage": 21.0,
    "vatName": "T40 21%",
    "status": [ "Processed", "PartlyInvoiced", "ToPurchaseOrder" ],
    "creationDate": "2024-01-25T16:56:54.377",
    "modificationDate": "2025-04-03T11:30:29.757"
  }
}
```

It is also possible to optionally specify an `extraItemFields` parameter in the request, similar to how this is done in the request of 10.3.1.

10.5 Add single order line

This method allows the caller to add additional order lines to a given sale order.

10.5.1 Request

```
POST /api/v1/sale/orders/22.425/items

{
  "orderLine": {
    "article": "55453",
    "quantity": 5.5,
    "salePrice": 12.50,
    "properties": [{
      "name": "Handling",
      "parameters": [
        { "name": "Borduren", "value": "True" },
        { "name": "Bedrukken", "value": "True" },
        { "name": "Foto", "value": <base64 encoded file content>, filename:
"myfile.jpg" }
      ]
    }]
  },
}

→

{
    "order": {
        "number": "SO/22.425",
        "customerNumber": "C/13174",
        "creationDate": "2024-01-25T16:15:19.447",
        "modificationDate": "2025-04-03T11:30:29.757"
        "startDate": "2022-11-10T00:00:00",
        "status": ["SentByFax", "ToPurchaseOrder"],
        "items": [
            {
                "number": "SO/22.425.1",
                "itemNumber": "CI/11258.14789",
                "name": "Item name",
                "quantity": 1.00,
                "name": "Item name",
                "unitCode": "kg",
                "salePrice": 17.8660,
                "vatPercentage": 21.00,
                "vatName": "T40 21%",
                "status": ["ToPurchaseOrder"],
                "creationDate": "2024-01-25T16:15:19.447",
                "modificationDate": "2025-04-03T11:30:29.757"
                "extraFields": {
```

```
                    "field1 ": 1.0
                },
                "properties": [
                    {
                        "name": "Handling",
                        "parameters": [
                            {
                                "name": "Bedrukken",
                                "value": "True",
                                "creationDate": "2024-01-25T16:15:19.447",
                                "modificationDate": "2025-04-03T11:30:29.757"
                            }
                        ]
                    }
                ]
            },
            …
        ]
    }
}
```

article	Article code for the given order line. This refers to the customer defined identification number for a given article ("artikel nummer"), not its Yerpa number.
quantity	Number of items to order
salePrice	Unit price (excluding VAT) of the item being ordered (optional – if omitted, Yerpa will look at the currently valid tender)
properties	Allows the caller to set a number of properties on the sale order item.
remark	A remark for the given order line (optional).

Please refer to chapter 13 (Properties and parameters) for more information on sending properties and parameters.

10.5.2 Response

The response will contain an object called `order` similar to the response when retrieving a full order (see 10.3.2).

10.6 Update single order line

This method allows the caller to retrieve the details of a given sale order line based on its number.

10.6.1 Request

```
POST /api/v1/sale/orders/24.642/items/1
{
  "quantity": 7
}

→
```

```
{
  "orderLine": {
    "number": "SO/24.642.1",
    "name": "KASSIE | O1 | Safety Jogger | Zwart | 39",
    "itemNumber": "CI/11259.3345",
    "quantity": 7.0,
    "unitCode": "Paar",
    "salePrice": 43.23,
    "properties": [
      {
        "name": "Handling",
        "parameters": [
          {
            "name": "Bedrukken",
            "value": "True",
            "creationDate": "2025-04-03T10:12:06.883",
            "modificationDate": "2025-04-03T10:12:09.23"
          }
        ]
      }
    ],
    "vatPercentage": 21.0,
    "vatName": "T40 21%",
    "status": [ "Processed", "PartlyInvoiced", "ToPurchaseOrder" ],
    "creationDate": "2024-01-25T16:56:54.377",
    "modificationDate": "2025-04-03T11:30:29.757"
  }
}
```


Currently the only value that can be modified is the `quantity` field of the order line.

10.6.2 Response

The response will repeat the entire order line information block.

10.7 Remove single order line

This method removes a given sale order line from its order based on its number.

10.7.1 Request

```
POST /api/v1/sale/orders/21.1433/items/1/delete
{
}

→

{
    "success": true
}
```

No request body needed. Do pay attention to the use of the /delete suffix in URL instead of using HTTP DELETE method.

10.7.2 Response

A single field `success` containing the value `true`.

11 Sale invoice methods

11.1 Update invoice payment status

11.1.1 Request

```
POST /api/v1/sale/invoices/21.1433/payment-status
{
    "loginHash": "c1c1d41fe2f24ce62a34b",
    "status": 1
}

→

{
    "invoiceNumber": "SI/21.1433",
    "status": 1
}
```

The invoice to update is identified by the number used in the URL. To update invoice SI/21.1433, the URL used should be /api/v1/sale/invoices/21.1433/update-payment – so without the SI/ prefix.

The `status` parameter is the payment status to set, where 0 means *not paid*, 1 means *partly paid* and 2 means *fully paid*.

11.1.2 Response

The response will echo the specified Yerpa invoice number (but including the SI/ prefix) and the newly set status.

12 Purchase delivery methods

12.1 Create purchase delivery

This method creates a new purchase delivery (and perhaps supplier) in Yerpa.

12.1.1 Request

```
POST /api/v1/purchase/deliveries/create
{
    "loginHash": "c1c1d41fe2f24ce62a34b",
    "supplier": {
        "number": "C/353",
        "ccode": "5334143634534",
    },
    "deliveryLines": [
        {
            "articleNumber": "CI/33.258",
            "articleCode": "55453",
            "quantity": 5.5,
            "purchasePrice": 12.50
        },
        ...
    ]
}

→

{
    "supplier": {
        "number": "C/353",
        "companyName": "Company A",
        "vat": "BE0665488896",
        "name": "Jan Peeters",
        "address": "Stationstraat 5",
        "postalCode": "BE 2000",
        "city": "Antwerpen",
        "country": "BE",
        "phone": "+32-3-6001234",
        "email": "jan.peeters@companya.be"
    },
    "deliveryNumber": "PD/19.23",
    "referenceSupplier": "Supplier reference"
}
```

We identify two major parameters that need to be sent to this method:

- `supplier`: either the `number` (= Yerpa ID for this supplier) of `ccode` (= company code) field should be passed along to select an existing supplier. It is currently not possible to create new

suppliers.

- `deliveryLines`: defines the content of this order
- `referenceSupplier` let you pass a supplier reference along with the order.

Delivery line properties:

articleNumber	Yerpa number for the item used in this delivery line.
articleCode	Article code for the item used in this delivery line. This refers to the customer defined identification number for a given article ("artikelnummer"), not its Yerpa number.
quantity	Number of items to be delivered
purchasePrice	Unit price (excluding VAT) of the item being delivered (optional)

Either the `articleNumber` or `articleCode` should be supplied to specify the item to be used.

12.1.2 Response

The response will contain information on the newly created or existing supplier, including his Yerpa number. It is advisable to store this number and reuse it later when creating new deliveries for the same supplier.

Apart from the supplier information, the response will also include a `deliveryNumber` field that contains the Yerpa number for this delivery. This could be interesting information to store, since it allows you to easily locate your delivery in Yerpa later on.

13 Properties and parameters

Some of the above-mentioned API methods allow the caller to create or update properties and parameters in the context of a certain record. For example, creating a sale order allows the caller to specify properties and parameters to attach to the individual order lines.

When transmitting those, the caller is supposed to send an array of property objects. Each property object contains a `name` field and a `parameters` array. In turn, this parameters array should contain parameter objects, each having a `name` and `value` field.

```
"properties": [{
    "name": "Handling",
    "parameters": [
        { "name": "Borduren", "value": "true" },
        { "name": "Bedrukken", "value": "true" },
        { "name": "Foto", "value": <base64 encoded file content>, filename:
"myfile.jpg" }
    ]
}]
```

The `value` field should contain a string value:

- For text parameters, the value will be treated as such.
- For number parameters, the value should be a string representation of said number, using a point as the decimal sign.
- For boolean parameters, the value should be either “true” or “1” for *true*. Any other value will be interpreted as *false*.
- For file parameters, there should be an extra field called `filename` which should contain the name of the file. If the `value` field contains data, it will be parsed as a Base-64 encoded string and a new file will be created with the given filename. If `value` is empty or omitted, the system will assume there is already a file with the given name stored in Yerpa, and a reference to this file will be created.

14 Status filtering

Some methods (notably the item search method for now) allow the caller to include a `status` field in the request body. This field is a comma-separated list of statuses that should be set on the record to include it in the search results. For example, the value “webshop,onhold” will display items that have both the Webshop and OnHold statuses.

It is also possible to invert these individual statuses by prefixing them with an exclamation point. For example, the value “webshop,!inactive” will only include the items that have the Webshop status and NOT the Inactive status.